Re-evaluating the revisioned S800 Dataset for Species Recognition: A Cross-Corpus Approach Using BioBERT

Master of Science Thesis
University of Turku
Department of Computing
Computer Science
2024
Harttu Toivonen

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

UNIVERSITY OF TURKU
Department of Computing

HARTTU TOIVONEN: Re-evaluating the revisioned S800 Dataset for Species Recognition: A Cross-Corpus Approach Using BioBERT

Master of Science Thesis, 180 p.
Computer Science
May 2024

The S800 dataset is utilized for identifying named entities in biomedicine. However, the S800 under-performs many other biomedical datasets designed for named entity recognition. Consequently, the S800 dataset was re-annotated. This Master's thesis focuses on a comparative evaluation of the new and old versions of the dataset from the perspective of species entities. Employing natural language processing and named entity recognition, the research aims to train a BioBERT-based model to recognize species in texts by leveraging context. The study uncovers inconsistencies in the old S800 dataset, particularly in the naming of species subtypes, affecting its performance. Comparing the new and old S800 datasets and contrasting them with the LINNAEUS dataset yielded results indicating an improvement in species recognition but also highlighted issues within the LINNAEUS dataset. Efforts were made to identify the dataset sections that produced the most errors, and species were examined by subcategories. Metrics were developed, potentially applicable to broader error analysis in biomedical texts.

Keywords: NLP, BERT, BioBERT, NER

TURUN YLIOPISTO
Tietotekniikan laitos

HARTTU TOIVONEN: Re-evaluating the revisioned S800 Dataset for Species Recognition: A Cross-Corpus Approach Using BioBERT

Pro gradu -tutkielma, 180 s.
Computer Science
Toukokuu 2024

S800 on aineisto, jota hyödynnetään nimettyjen entiteettien löytämiseen biolääketieteessä. S800 toimii kuitenkin huonommin kuin monet muut nimettyjen entiteettien tunnistamiseen tarkoitetut biolääketieteelliset aineistot. Tämän takia S800-aineisto annotointiin uudestaan. Tässä maisterintutkielmassa keskitytään sen uuden ja vanhan aineiston keskinäiseen arviointiin lajientiteettien näkökulmasta. Luonnollista kielenkäsittelyä ja nimettyjen entiteettien tunnistusta käyttäen tutkimuksen tavoitteena on kouluttaa BioBERT-pohjainen malli tunnistamaan lajit tekstistä hyödyntäen kontekstia. Tutkimus paljastaa epäjohdonmukaisuuksia vanhasta S800-aineistossa, erityisesti lajien alatyyppien nimeämisessä, mikä vaikuttaa sen suorituskykyyn. Vertaillessa uutta ja vanhaa S800-aineistoa ja vertaamalla sitä LINNAEUS-aineistoon, saimme tuloksia, jotka viittaavat lajien tunnistuksen parantumiseen, mutta myös viittavaat ongelmiin LINNAUES-aineistossa. Lisäksi pyrittiin löytämään aineistosta niitä osia, jotka tuottivat kaikista eniten virheitä ja lajeja tutkittiin myös alakategorioittain. Tähän laadittiin metriikoita, joita voi mahdollisesti hyödyntää myös laajemmin virheanalyysistä biolääketieteellisessä tekstissä.

Asiasanat: NLP, BERT, BioBERT, NER

Contents

1 Introduction

This Master's thesis focuses on the re-evaluation of the S800 dataset, through which its annotation was refined, resulting in a qualitatively improved dataset, and presents suggestions for its future development. The S800 data set `https://species.jensenlab.org/` is used for machine learning in the biomedical field. The data set consists of 800 PubMed abstracts, of which species entities are annotated. In the data set, there are numerous different annotations groups such as genera, families and orders, but in this study, we focus only on species. The goal of the model trained from the S800 dataset is to find species in scientific texts.

Species recognition is important for several reasons such as:

- As a basis for other NLP tasks such as relation extraction.
- Helps in compiling larger databases about different species including their habitats, behavior, and interactions
- Interdisciplinarity: species recognition facilitates the use of information across different fields of research such as ecology and genetics.
- Given the high volume of scientific publications today, it is important that species can be efficiently identified from them.
- It can also be used to improve public health, for example, by monitoring certain diseases (spread through mosquitoes and ticks)

In the S800, there is a reference to the corresponding NCBI taxonomy name. The field that deals with these kinds of problems in computer science and, more specifically, machine learning is known as Natural Language Processing. The task of finding species in a text falls into its subtype problem, Named Entity Recognition. For finding species entities, there are various approaches. In a naive approach, one must have a dictionary with the NCBI taxonomy name and its variants, misspellings, common names etc. SPECIES in https://species.jensenlab.org/ is this sort of tool.

This is quite straight forward approach but requires a lot of work for mapping all the possible variants of how a specific species can be called. Moreover, the list needs to be updated over and over again as the field progress and language evolves. In this study, we use a different approach. Instead of making strict rules and keywords for finding species, we try to train a general language model that looks at the text and then, based on the full context, tries to learn whether a particular location in the text is species or not. For the task, we use a model from Google, the BERT model, which is a general language model that the company has trained over a long period of time and with exclusive computer time. The model is not the original BERT but a slightly modified model known as BioBERT, in which the training has been continued to suit specifically biomedical text. This study aims not to find the NCBI taxonomy identifier for a species mentioned in the text but rather to find its location accurately.

This study was inspired by the fact that recent development in NLP and machine learning, in general, has made many of the tasks that were previously hard, now feasible. Specifically, in NER-field data sets for recognizing chemicals (CHEMD-NER), cell line names (GELLUS) yield good results. However, the NER task for species recognition lacks behind. There are two data sets for species recognition, the S800 and the LINNAEUS. In our experiments where we applied BioBERT for various data sets, we found out the S800 was constantly giving much more poor

results than the rest. When examining the data set, it became clear that it was not very consistent. This became especially apparent in naming subtypes of species, and strains, as their naming conventions vary a lot, and names can be quite lengthy.

There are several challenges in building species recognition systems, such as

- Ambiguity in common names: many species have names that are common words in english, such as Bass.
- Polysemia: Some terms in biology have several meanings depending to the context. A term may refer to a genus or specific species name in another context.
- Domain-Specific Language and Jargon: Scientific texts often use specialized vocabulary and jargon, that can be difficult for NLP systems not specifically trained on biological texts to understand and process effectively.
- Multilingual, Cross-lingual Issues and Variation in Naming Conventions: Direct translation may not always exist so species recognition systems need to deal with names and descriptions in multiple languages. Also abbreviations and subfield dependant naming conventions complicate recognition effort.
- Scale and diversity of life: There are millions of species, many of which are poorly documented or not even discovered yet. Developing a system capable of accurately recognize and differentiate between them is challenging.
- Data sparsity and quality issues: There are only a few annotated datasets for training recognition models for species.

This thesis primarily addresses quality issues but also indirectly tackles the aforementioned challenges by generalizing to new species.

This thesis focuses on a specific version of this new the S800 data set, and it is dated 10.11.20 (ISO). Training and testing a data set is known as an in-corpus

study. In addition, we combined the S800 data set with the LINNAEUS and tested them against each other's data sets. This is known as a cross-corpus and combined-cross-corpus study.

We are particularly interested in the following research questions:

1. How much improvement re-evaluation of the S800 dataset gives? We do this on in-coprus, cross-corpus and combined-cross-corpus corpora.

2. How much training set size contribute to performance. Is the model getting better with more training data?

3. Are there categories of species which are particularly difficult to recognise?

4. Is looking merely at overall F-score reasonable and what can be found by more extensive evaluation strategies?

The structure of this thesis is the following: We first go through some historical background of deep learning and go over the very essence of the BERT model, neural networks. Then we focus on different NLP methods and introduce the data sets. In the second part, we first go over the development runs in which the model is trained and look at how the model performs on them and then delve deeper into the test runs, which determine the final score given to each run. We will take a look at the mistakes the model makes and then try to analyze further with different methods on how we can spot those errors. We will draw together the results and discuss further studies in the final part.

1.1 Neural Networks

Neural networks are pivotal in modern Natural Language Processing (NLP) for many reasons. They have led to breakthroughs in tasks like language modeling and machine translation by learning complex language structures directly from data. They are a backbone of deep learning for architectures such as CNNs, RNNs, and transformers and thus we shall take a look how they work.

1.1.1 A brief history and evolution of Neural Networks

In the '50s: Birth, hype and winter

The study of Neural Networks is a subfield of the study of Artificial Intelligence. Neural networks are currently the most widely used and promising form of AI. Its roots date back at least to the 1950s. In 1958 at Cornell Aeronautical Laboratory, Frank Rosenblatt invented an algorithm called the perceptron (Rosenblatt)[1]. Its first implementation was in image recognition for the IBM 704. We will take a brief look at the development of Neural networks, and we will see that, in some sense, relatively little has changed in the basic architecture.

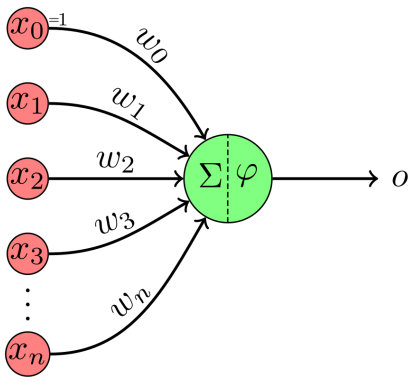

Figure 1.1: The perceptron: Based on inputs $x_1, x_2, x_3, ...$ the perceptron classifies the data point. Source: Wikipedia, Author: *MartinThoma*, Licence: CC0

The perceptron(image 1.1) is a linear binary classifier. It means that for a given

input, it is able to predict whether a particular data point belongs to class A or B. Let's take a look at an example image 1.2.

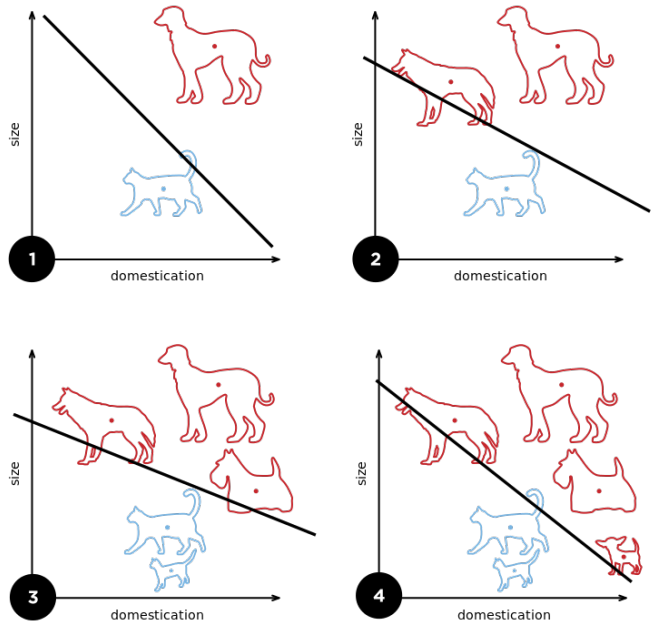

Figure 1.2: The perceptron adjusts itself when more data points are added. Source: Wikipedia, Author: *Elizabeth Goodspeed*, Licence: CC BY-SA 4.0

In this example perceptron is trained to be able to distinguish an animal as a cat or a dog based on its size and level of domestication(how far is the process of taming the animal). As more data points are added the perceptron adjusts itself to better predict all of the data points. The mathematics of the perceptron is quite simple. It associates a weight with the inputs and sums their product. Then a bias term is added, and the result is compared to a threshold, zero in this case. If the overall product is positive, the sample belongs to class A and if it is negative it belongs to class B. We will look into mathematics in more detail later in this chapter.

The perceptron soon gained attention and hype as it was predicted it would soon replace humans in many complicated fields, such as machine translation. However, it did not take long when Minsky and Papert (1969)[2] discovered one evident weakness. Most notable is the observation that a single perceptron is not able to tackle

the XOR problem, where the data points are not separable by a single line.

This notion, even though it does not mean that a collection of perceptrons wouldn't be able to handle the XOR(image 1.3), proved that perceptrons are not omnipotent and led to a widespread freeze of funding and publications. The era followed by that event is known as AI winter.

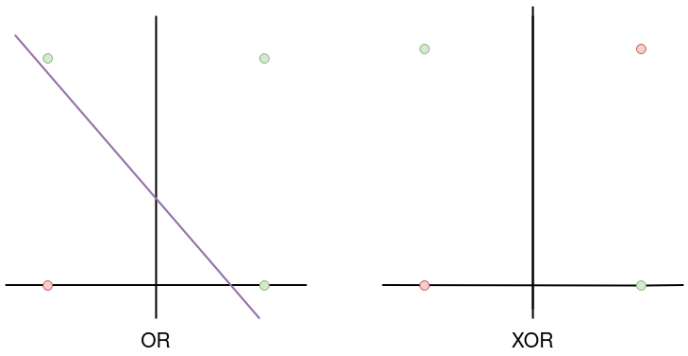

Figure 1.3: A single perceptron is not able to emulate the XOR gate. (Image: Author's own creation)

Late 80's, early 90's and onwards

There was another boom in the neural network field in the 1980 and early 1990. Instead of using just a single neuron, many neurons and an additional hidden layer were used(image 1.4). Multiple hidden layers make it possible for the network to learn features inside the layers. For example, for image data, the first layers can learn what a line or circle and the layers after can learn more complex forms such as a face or a car from those features.

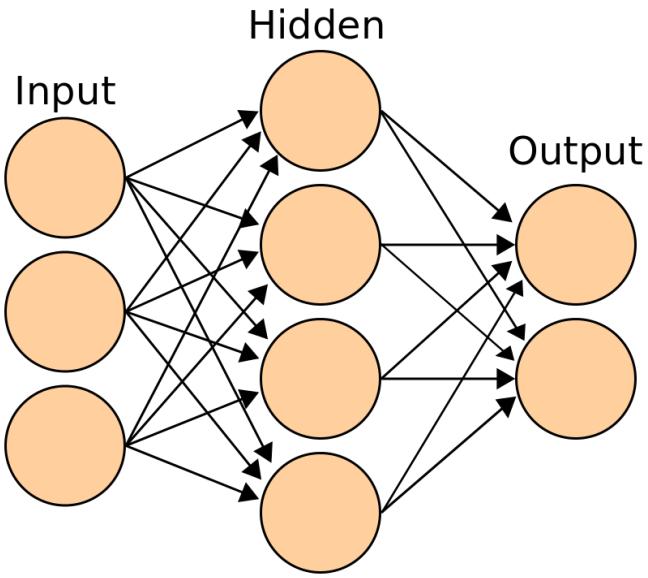

Figure 1.4: Neural networks have 1 hidden layer. Source: Wikipedia, Author: *Cburnett*, Licence: CC BY-SA 3.0 CC

One notable example from the late 1980s are Yann LeCun's LeNet which is able to understand handwritten digits(image 1.5). It uses a special type of layers called convolutional neural networks. The idea is to use several patterns to go over the image and learn regional features from the image. For example, one pattern can learn to distinguish a horizontal line and another vertical line. Then those patterns will work anywhere in the image.

Figure 1.5: LeNet is able to translate handwritten digits to numbers. A task that is very hard to achieve with traditional rule-based programming. Source: Wikipedia, Author: *Suvanjanprasai*, Licence: CC BY-SA 4.0 CC

Modern times

The perceptron-based approaches were not however forgotten, researchers continued to work with them and in the 1990s there was another hype and in 2010 another. The new era is thanks to the improved computation power, developed alongside the gaming industry specifically culminating in Graphical Processing Units and data availability in open internet resources such as the Wikipedia.

Inner working of perceptrons and neurons

There has been a debate going on how much neural networks have actually got to do with the brain(image 1.6). Clearly, there has been some motivation in their invention. If we take a look at a brain cell and a perceptron, we can see the resemblance: In a brain cell signals are passed onto dendrites. In the perceptron that would be the data points x_i and their corresponding weights w_i. In a brain cell, the signals from dendrites are collected in the nucleus and if the overall signal reaches some limit, an electric signal is passed on via an axon. The perception, equivalent would be to compare the sum to a threshold.

In mathematical notation for the perceptron would be

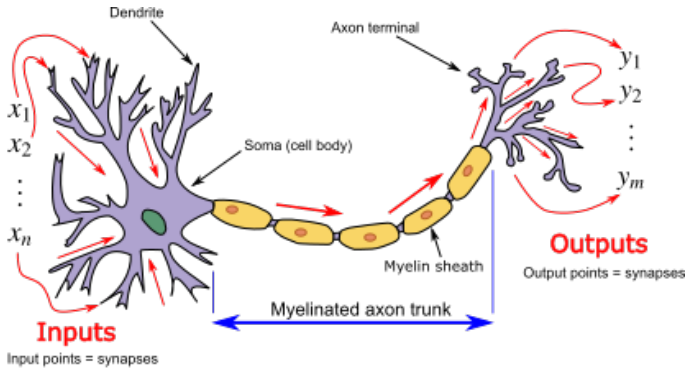

Figure 1.6: The perceptron has some resemblance to a brain cell. Source: Wikipedia, Author: *Egm4313.s12*, Licence: CC BY-SA 4.0 CC

$$f(\boldsymbol{x}) = \begin{cases} 1 & \sum_{j=1}^{n} x_j w_j + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

We denote a vector x_i by making it bold: $\boldsymbol{x}$. The sum part can be replaced by the dot product: $\boldsymbol{x} \cdot \boldsymbol{w} > 0$ where x and w are vectors.

The analogy is more detail in the following: Each element x_i of an n-dimensional vector $\boldsymbol{x}$ is associated with a weight to the input w_i and a bias term b/n. The product is passed forward $x_i w_i \ + b/n$. All of the signals derived from the *dendrites* are collected in the center - *nucleus* - of the perceptron, in which they are summed up $\sum_{x=0}^{n} x_i w_i + b$. The perceptron is associated with a term called an activation function. It means that the output of the perceptron is not just the linear dot product, but in this case a step function. The perceptron is a special case where the activation function is a step function. In a more general case, it is usually a non-linear function, such as tanh, relu. We will take a look at those later.

Beyond perceptron

So far, we have examined only the perceptron. As mentioned earlier, the perceptron cannot cope with the XOR-gate problem. However, that is not to say that a collection of them wouldn't be able to handle it. And indeed, the history of neural

networks is mostly about the discovering of the perceptron and then many of them in layers(image 1.7). There are also different activation functions, but the main idea hasn't changed a lot.

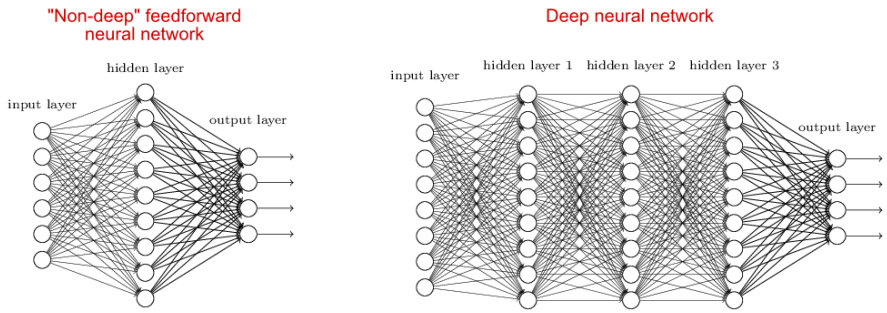

Figure 1.7: Modern neural networks are essentially layers of simple perceptrons with activation functions stacked together. Source: `stats.stackexchange.com`, Author: *amoeba*, Licence: CC BY-SA 3.0 CC

1.1.2 Mathematics of Neural Networks

Let us look briefly at the mathematical backgrounds of neural networks. We will focus on the classification task because the main problem of our interest in the named entity recognition for the S800 is, at its heart, a classification problem.

Classification task

In machine learning regression model tries to predict continuous values. For example, given features from a neighbourhood such as the distance to a shop, to a school, to a kinder garden and from a house such as a number of rooms, year of building etc., predict the price of the house. The other main type of problem is the classification problem. Given a set of features, predict how likely a sample belongs to class A, B or C. One of the most famous problems is the Iris dataset problem. Iris dataset, introduced by Ronald Fisher, contains measurements of three types of Iris flour 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris

versicolor)[3]. Each sample belongs to a set of four species and has four features: the length and the width of the sepals and petals, in centimetres. The goal is to build a classifier to decide, given a set of features, which type of Iris is in question.

We will start with a small example to introduce the concepts we require to do machine learning. The example is not supposed to be working well and can be referred to as a toy example.

Example: Movie review classifier

Based on its context, let us try to predict whether a movie review was good or bad. This is a common introductory problem in machine learning, and the dataset is called the IMDB data set. It consists of examples such as:

A: *If you like adult comedy cartoons, like South Park, then this is nearly a similar format about the small adventures of three teenage girls at Bromwell High. Keisha, Natella and Latrina have given exploding sweets and behaved like bitches, I think Keisha is a good leader. There are also small stories going on with the teachers of the school. There's the idiotic principal, Mr. Bip, the nervous Maths teacher and many others. The cast is also fantastic, Lenny Henry's Gina Yashere, East-Enders Chrissie Watts, Tracy-Ann Oberman, Smack The Pony's Doon Mackichan, Dead Ringers' Mark Perry and Blunder's Nina Conti. I didn't know this came from Canada, but it is very good. Very good!*

being a positive review and

B: *This movie was terrible. at first i just read the plot summary and it looked OK, so i watched it. The acting was TERRIBLE. it was like the actor were almost camera shy. everything seemed fake. i feel bad for Edward Furlong, terminator 2*

was my favorite a few years ago.. I've watched it at least 20 times.... the plot was also crap. the writers were probably sleep deprived when they came up with the lines. on the plus side, it's the good kind of bad movie. the one you keep watching just to see how much worst could it will get, so that later you can tell other people how you couldn't believe how terrible the movie was.i think everybody should watch this, so that then we could appreciate better other, REAL, movies.

being a negative review.

How would we go about building a classifier for this task? Since this is textual data the input of the classifier should be text in some form. The simplest way would be to feed all the words in a review for the classifier. That way the classifier would have the maximum information from a sample to decide. However, that makes things a bit too complicated right from the start. So for the sake of simplicity, let us only look for certain words. Namely: *good*, *bad*, *fantastic* and *terrible*. Following this convention sample A would be decoded as [1,0,1,0] because it has the words *good* and *fantastic*, and sample B would be [1,1,0,1]. Of course, the above words were selected only for the purposes of classifying this two text but the idea of collecting such a list of words which could serve as good classifiers is called feature selection. We have now transformed the text samples into features that will be our neural network's input. What would be the outputs? In this case, we simply consider each sample either positive or negative, hence the output should only have two neurons, one for positive and the other for negative.

How will the network actually work? When we feed a positive sample to the network we would like that the positive neuron has a bigger value in the end than the negative neuron. So feeding the sample A [1,0,1,0] would ideally result [1,0] as the output and the sample B [1,1,0,1] the result [0,1].

The learning objective

In an ideal world sample A would map to positive and sample B to negative. But what if that wasn't the case, there needs to be a way to tell the model is not working correctly and teach the model to do better. For that, we have to be able to tell the model how well it is doing. That is done with a loss function. It basically gives a score of how well the model is doing. Suppose we have four samples from the IMDB dataset: A (pos), B (neg), C (neg) and D (pos). The most intuitive scoring system would be to give a point for every right answer. So if our model predicted [pos, neg, pos, pos], it would have scored 3 out of 4.

sample	A	B	C	D
right	pos	neg	neg	pos
predicted	pos	neg	pos	pos
score	1	1	0	1

However, we will turn these concepts upside down. Rather than thinking that the higher the number, the better the result, we will think the lower the number, the better the result. So we will measure **loss** and hope it to be as small as possible.

sample	A	B	C	D
right	pos	neg	neg	pos
predicted	pos	neg	pos	pos
loss	0	0	1	0

The loss function

This concept of ranking the performance of the model in terms of its loss can be expressed in mathematical form. The approach we introduced in the previous section is known as the zero-one loss function:

$$L_{0/1}(h) = \frac{1}{n} \sum_{i=1}^{n} \delta_{h(x_i) \neq y_i} \tag{1.1}$$

where the delta δ is 1 if the network produced right answer, and 0 if it did not. In mathematical terms:

$$\delta_{h(x_i)\neq y_i} = \begin{cases} 1 & h(x_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

We use the symbol h in the formula. In the theory of machine learning the term hypothesis space is used. It refers to a set of all the possible hypotheses and the goal of machine learning is to find the best hypothesis from that set. That may sound a bit abstract, so the hypothesis in this context can be thought of as a specific type of neural network characterized by its number of hidden layers, inputs, outputs, etc. Our goal is to find the parameters which give us the lowest loss.

The zero-one loss function is quite simple and easy to understand - although the mathematical formulation might be a bit intimidating at first. However it is only applicable for a discrete variable, that is for the classification tasks. For a continuous variable, that is regression we must use another type of loss function. One of the most widely used is squared loss.

$$L_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2 \tag{1.3}$$

The squared error sums up the squared distance between the predicted values and observation. Squaring the difference of $h(x_i)$ and y_i makes the result positive, but it also exaggerates the larger difference. To make it more linear, squaring can be replaced by simply taking the absolute values between prediction and observation and taking the average.

$$L_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n}|(h(x_i) - y_i)| \tag{1.4}$$

Let's take a brief look at the term hypothesis. In the image (figure 1.8) below, there is a dataset in red and a line in black. The problem is a linear regression problem and hence we use the squared sum error loss function. A line is fitted onto

a set of points (red). Term k, the slope, is the only free variable. On the left, the squared less is projected as a function of slope term k. we can see, that the best value for the slope is around 1. At this point, the loss is at its lowest. Moving to the right and to the left makes the loss increase rapidly.

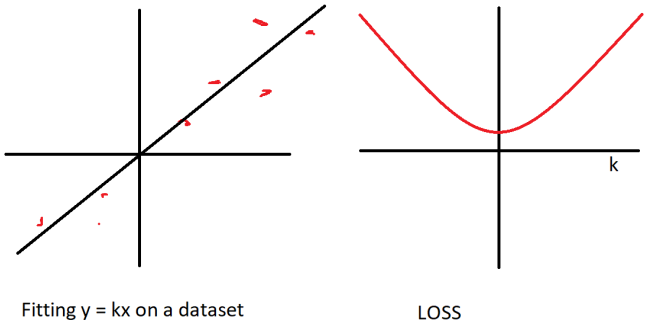

Figure 1.8: Linear regression problem. A line is fitted onto a set of points (red). Term k, the slope, is the only free variable. On the left, the squared less is projected as a function of slope term k. (Image: Author's own creation)

Now we know how to rank the classifiers. We choose the one that has the smallest loss. But how exactly can we do that? Before dwelling into that question we shall briefly go over some of the terms we have introduced earlier but now with a mathematical perspective.

The Forward pass

Pushing a sample point **x** as input through the network and reading its output $h((x)$ is known as a forward pass. Let us examine what happens in a node in image 1.9.

Let the inputs to a neuron be denoted by x_i, where i goes from 1 to the number of inputs. The inputs to the neuron are associated with weights. The weighted sum is superposition of input, that is $w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n$. In vector notation, that sum is interpreted as dot product, and can be written as $w^T * x$, where bold(w) and bold(x) are column vectors. The product is then forwarded to an activation function. The idea of an activation function is to make the network

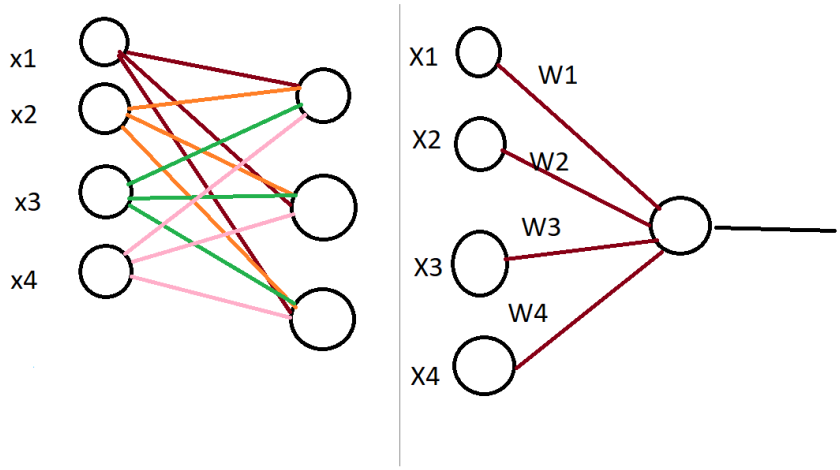

Figure 1.9: On the left, 4 inputs. 1 hidden layer. 2 outputs. On the right, a single neuron examined. (Image: Author's own creation)

non-linear. Otherwise, a network with many hidden layers would collapse into the very simple network because of the linearity. Note that linear mappings are only able to rotate and scale space, and for that they won't be good for complicated problems.

There are several alternatives to activation functions most commonly used are sigmoid, tanh and relu. Let us consider the sigmoid function.

We will take a look at the alternatives and look at the interaction between them

The sigmoid function (image 1.10) squishes numbers to a range between 0 and 1. At $x = 0, f = 0.5$, and from there, it rapidly converges to 1 when moving to right and to 0 when moving to left. This activation function is important for various reasons. First of all, it keeps the values that the cell can output in a certain range. Values between 0 and 1 can be interpreted as probabilities. For example for a binary classification problem, this property is especially handy as the last step of a neural network because it provides a neat way to decide whether a value is a given class or not. There are also other activation functions. One of the most common at the time of the writing is RectifiedLinearUnit. ReLU (image 1.11). It is simply a function that equals to zero for $x < 0$ and a linear function $f(x) = x$ for $x \geq 0$.

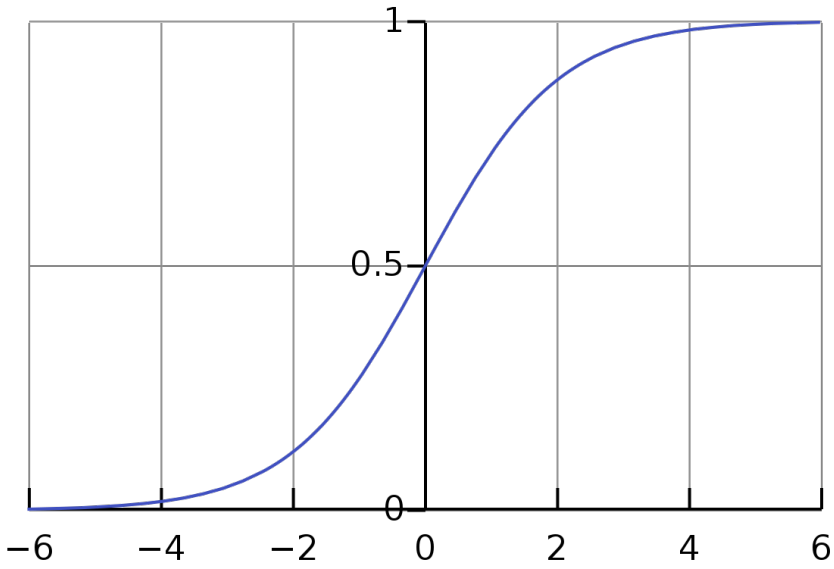

Figure 1.10: The sigmoid function. Source: Wikipedia, Licence: Public domain

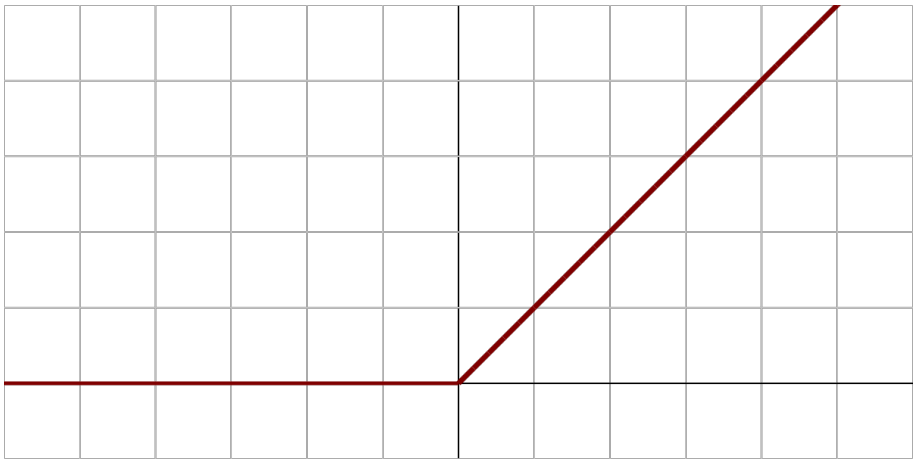

Figure 1.11: The ReLU function. (Image: Author's own creation)

Backpropagation

In order to make the network learn, we must alter its parameters so that the loss function is minimized. A neural network can contain hundreds of thousands of interconnected parameters, so at first glance, it would seem impossible to tweak them together so that the loss function is really minimized. However, since a neural network consists of continuous derivable functions, the chain role applies to them, and we can use it to find the gradient to minimize the loss function. In the learning,

we need to take a small step, that is, the learning rate, into the direction of the gradient. If the step is too large, we might accidentally make the loss higher. So we must keep track of the learning rate. Some methods adjust the learning rate, so it will diminish at the end of the training.

The backpropagation method does not know whether the gradient is heading to the global or local minimum(image 1.12). If it sticks in the local minimum, it can never reach the global minimum. However, there are different techniques to avoid this scenario. One simple method is to run the backpropagation algorithm several times and start it in different places.

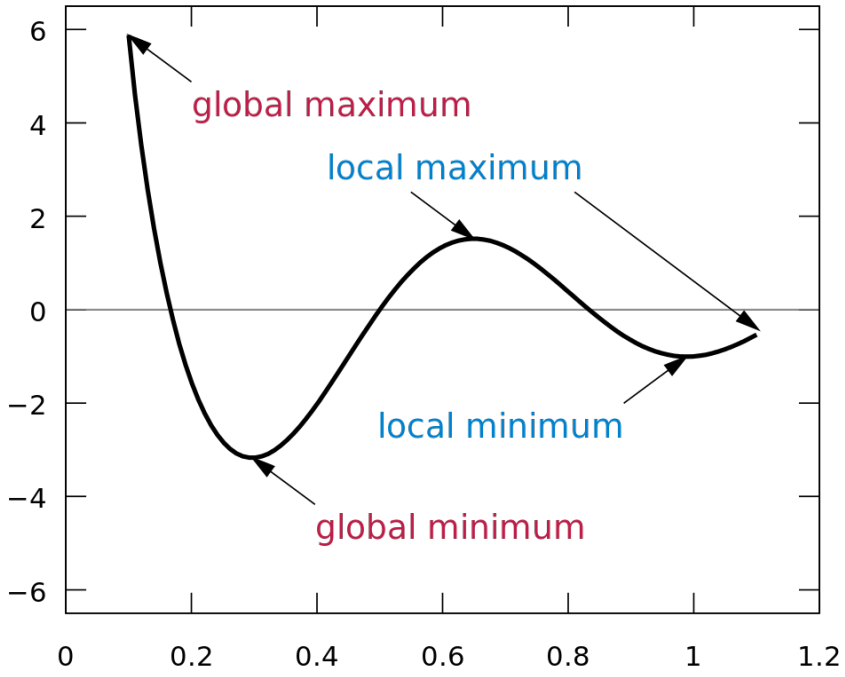

Figure 1.12: Loss function example. Source: Wikipedia, Author: *KSmrq*, Licence: GFDL 1.2

Data set splits

A neural network algorithm uses a dataset to adjust its parameters. As it can be very large, and have a very large number of parameters, it can learn the data set

by heart. In Figure 1.13 we can see that a too complex model learns the data very well, and introduces a high order polynomial, even though a simpler linear model would yield better results. This is called **overfitting**. Instead of adapting well to a particular data set, and learning it by heart, we want the model to generalize instead. To avoid overfitting, we divide a data set into three different parts:

1. **Training data** - used in the training process. This is the data set that the gradient descent uses to minimize error

2. **Development data** - used to compare different models during training

3. **Test data** - To give a final score to the model

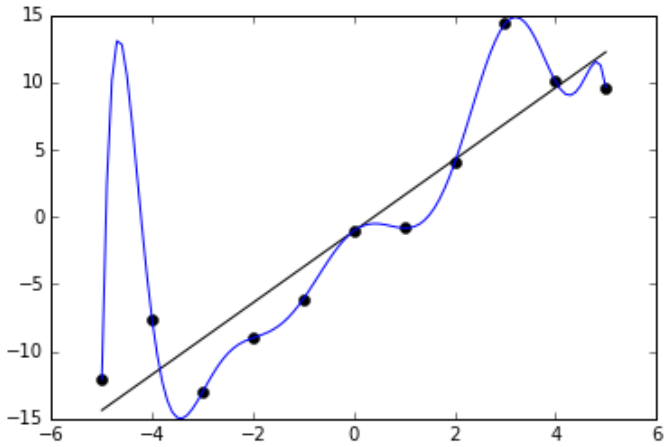

Figure 1.13: Complex model (blue) learns the data better than a simple model (black) but does not generalize well on unseen data. Source: Wikipedia, Author: *Ghiles*, Licence: CC BY-SA 4.0

In training, we use training data and our objective is to make the error between real outputs and predicted outputs as small as possible. As discussed previously, backpropagation gives us a means to adjust model parameters, so that predicted output get closer to real outputs. In the training process, we can at specific intervals evaluate the model on development data and see if its performance gets better also on external data. When the model has lost its ability to generalize the error of training

data still decreases but the error of evaluation data may start to increase, at this point we might stop the training process to try to avoid overfitting. (see Figure 1.14. This approach is called **early-stopping**. Usually, we alter the parameters of a model and run the training and evaluation process on each parameter combination, and, from those, choose the best performing model. The final evaluation score is evaluated with the test data set.

This ends up our discussion of the basic concepts of a neural network. In the next chapter, we shall take a closer look at the specific model which we are going to use in this study.

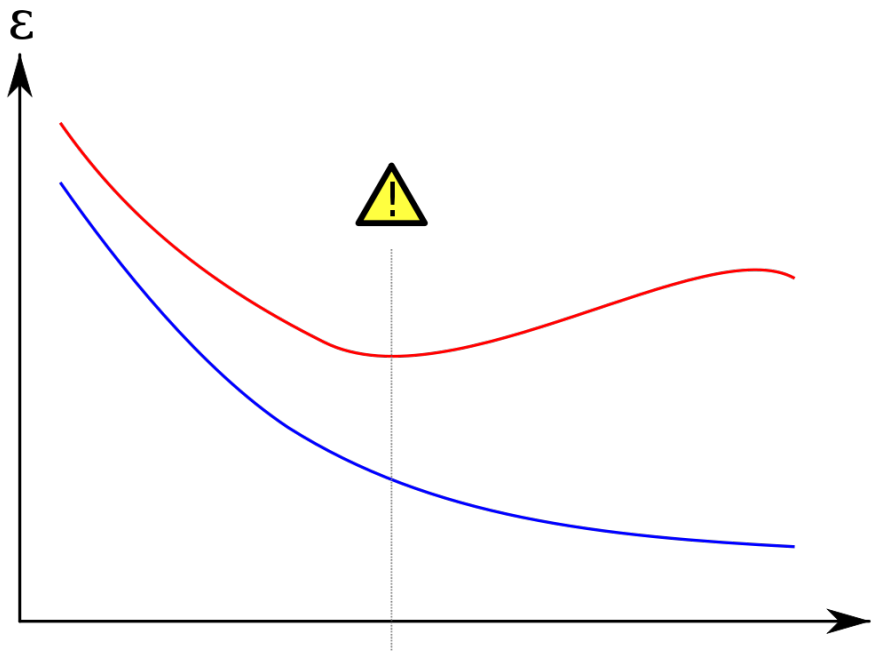

Figure 1.14: In order to avoid overfitting, early stopping can be used to stop the learning process once the loss function of evaluation (red curve) data does not improve, even if the loss value of training data (blue curve) would decrease. Source: Wikipedia, Author: *Gringer*, Licence: CC BY-SA 3.0

2 Methods

3 Natural Language Processing

3.1 Natural Language Processing

To process language inside a computer program, we need to represent the data in such a way that it is understandable to a computer. In the case of language, this task is quite complicated. If you are familiar with digital images, you may know that an image is just a matrix of values of light intensity.

This type of data can be fed into the neural network quite easily. Just turning the matrix into a long vector that is stacking its columns one after the other. Therefore $M \times N$ image (matrix) will just be $M \times N$ vector. Also, the learning process inside the network is somewhat understandable. Data will flow through layers that try to learn general features of the images, such as horizontal and vertical lines and from those, build more and more complex shapes such as circles.

However, this somewhat straightforward technique is not applicable to languages. Instead, we need to consider how we can represent words and their relations inside the model. We shall build our understanding from early models to modern inventions. We will not be using simple techniques, but anyway, go through them to be able to understand better the newer methods.

One option to represent words as input data is to take into account all the words *or at least the most frequent ones* in a specific language, and then for each document, input data in question, tell the model if a specific word is found in the document or

not. This is known as bag-of-words.

One-hot encoded vector is a vector which has values either 0 or 1 in its elements, as $[0, 0, 1, 0, ..., 1, 0]$. Bag-of-words model takes into account all the words in a given data set. These words form a vector, and each document in the data set is encoded by one-hot encoding the vector.

As an example, the IMDB data set contains movie reviews, and it is freely available to be used as a machine learning data set. To encode a document, the following procedure can be used:

1. **Vocabulary Compilation:** Go over all the words in our data set. Count how many distinct words are there and what are the most common words. Let's say there are approximately 100 000 words and the most common being "the", "a", "and" "of", "to". Do not take into account all the possible words but let us say 20 000 most common ones.

2. **One-hot Encoding:** Form a vector that has 20 000 dimensions and denotes each word with one 1 and the rest being 0's. That is one-hot encoding. E.g a being $[1, 0, 0, 0, ..., 0]$ and the being $[0, 1, 0, 0, 0, ..., 0]$.

3. **Document Encoding:** For a given document. Discard all words but those belonging to the most common 20 000 words group. Take words one by one and sum their corresponding one-hot encoding vectors.

4. **Vector Representation:** Now, a document is turned into a 20 000 dimensional vector constituting zeros and ones.

An example review from the IMDB data set: Probably my all-time favourite movie, a story of selflessness, sacrifice and dedication to a noble cause, but it's not preachy or boring. ... Could be $[1, 0, 0, 1, 0, 1, ...,]$, where a being $[1, 0, 0, 0, ..., 0]$, my being $[0, 0, 0, 1, 0, 0, 0, ...0]$ etc. As a side note, there may also be steps prior to this transformation, such as finding the root words for a word. For example: consult,

consultant, consulting, consultative, consultants, consulting... will all turn into the word consult.

The downside of this simple technique is that the words have no relationship in the representation. So the following data samples would produce exactly the same data encodings:

- (**A**) being *Sheila hates Justin, but Justin loves Sheila.* and
- (**B**) being *Justin hates Sheila, but Sheila loves Justin.*

That's because in both sentences words Justin, Sheila, hates and loves would be tagged to one and everything else to zero. So bag-of-words model is only able to get a grasp of the words but not the relations between them.

There are a few different techniques that try to address these issues. One of them is known as **n-grams**, which instead of taking into account only singular words, looks at pairs of words. For example, in the previous example, there would be pairs such as

- **A**: (*Sheila*,*hates*), (*hates*,*Justin*), (*Justin*,*but*), ...
- **B**: (*Justin*, *hates*), (*hates*,*Sheila*) ...

However, in this case, even pairs of 2 words can not capture the hate notion in this context. But instead, we would need 3-grams: (*Sheila*,*hates*,*Justin*), (*hates*,*Justin*,*but*), etc. The n-gram model captures word order, but it will suffer also from the curse of dimensionality *that is, having a very large vector sizes* since in a real data set, the number of all possible n-grams will grow rapidly.

The earliest reference to the concept of bag-of-words dates back to the 1950s (Harris). The Bag-of-word model does not consider whether a word is present many times or only once. Nor does it into account whether a word is rare in a data set. To

address these issues, there is a method called TF-IDF, which works similarly to bag-of-words but also takes into account the rarity and frequency of words. Even with frequency information added, bag-of-words models suffer from dimensions. Bag-of-words vectors tend to be huge, and most of the values are zeros. Of course, we do not have to consider all the words in a corpus, but still, the vectors will be quite sparse. Image a Twitter tweet represented in bag-of-words. Only 180 ones at maximum and tens of thousands of zeros. To address this problem, there is a technique known as word embeddings.

3.1.1 Word embeddings

Word embedding tries to represent words that have similar meanings close together and distant from words that have the opposite meaning. Think of the following sentences:

- A = "This movie was great"
- B = "This movie was superb"

In the one-hot-encoded world we would make a vocabulary of 5 words (this, movie, was, great and superb) and represent the sentences as A = $[1, 1, 1, 1, 0]$ and B = $[1, 1, 1, 0, 1]$. However, the terms great and superb are almost the same. They could be exchanged, and the sentences' meanings would not alter much. However, with bag-of-words, they would live in different dimensions and have no connection. We would like similar words to be close to each other in the feature space. One approach to deal with this issue is cosine similarity.

$$Cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{3.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are vectors and θ is the angle between them.

Cosine similarity captures how close are two vectors in the feature space. If they are perpendicular, they don't share similar dimensions, and their dot product is 0. And likewise, if they are pointing in the same direction, their dot product is close to 1. If they are pointing in the opposite direction cross-product will be close to -1. This idea was first introduced in 2013[4].

One of the most well-known examples of this might be the following: Take the words Madrid and Spain and subtract them: vec(*Madrid*) - vec(*Spain*). Then add to that word France: vec(*Madrid*) - vec(*Spain*) + vec(*France*). What would be the results? vec(*Paris*). This indicates that the new word vectors have a base that can encode meanings such as capital and country (see Figure 3.1). The main advantage of these kinds of approaches is that we can apply mathematics to the words and can examine which words are close to each other.

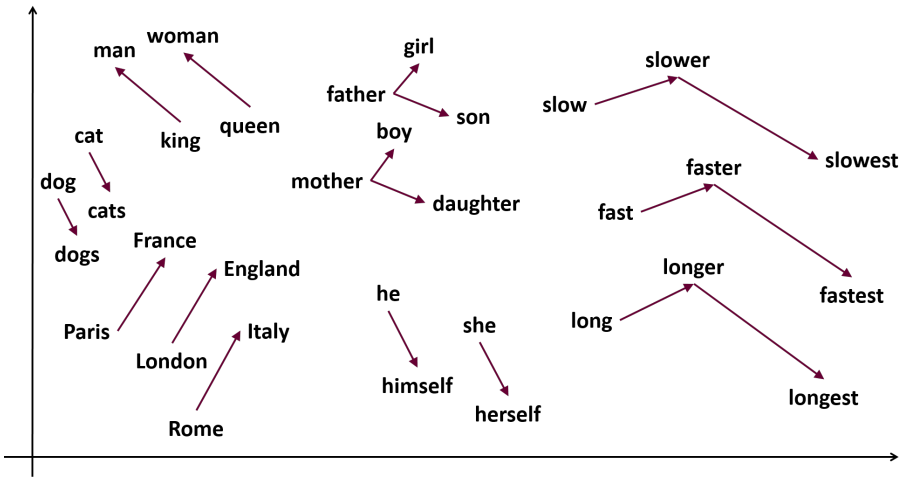

Figure 3.1: Embedding space. Source: `https://blog.csdn.net/zzulp/article/details/79379960`, Author: Unknown, Licence: CC BY-SA 3.0

A reasonable question is how one gets the encoded vectors in the first place. Let us think of the following sentence, *I believe people are good at heart*, and the context of the word *good*. Possible pairs of context are (good,people), (good,are), (good,at) and (good,heart). If we change the word *people* in the sentence to *humans*, the

meaning of the sentence does not change, and if we have lots of text, the probability of having some context for similar words, like *humans* and *people*, will be high.

One way to achieve this is to use encoder-decoder architecture on image 3.2. It works by inputting a word and trying to output the same word at the end. There is a hidden layer in the middle of the network, which is smaller than the input layer. When the network learns, it has to try to generalize the word information because it has to decode the information from a smaller layer.

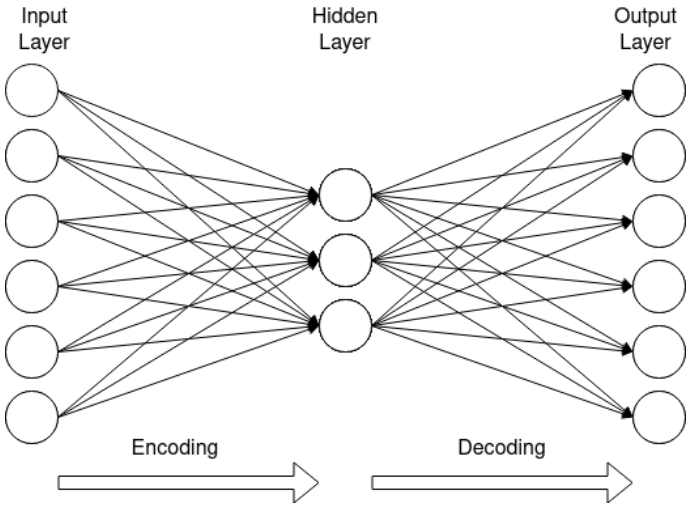

Figure 3.2: Encoder-decoder architecture: Input data (left) traverses through the center on its way to output (right). The center layer has to squeeze information, and it can be used to learn word embeddings. (Image: Author's own creation)

So the encoder-decoder architecture works by squeezing the information through the smaller intermediate layer, and thus the learning process forces the network to learn better representations. That is to try to find new bases on which the information can be represented. However, that doesn't mean that the new base vectors will automatically have a sense of gender or the idea of capital in the sense that humans understand them. But the *human base* can be found by subtracting two similar words.

There is also one huge advantage when using word embeddings. This is the idea that the embeddings are reusable and can be used in various NLP projects.

The pre-training process needs to be done only once, so there will be huge savings in hardware and time reduction. Pre-trained word embeddings offer a significant improvement over embeddings that are learned from scratch[5]. The idea of using pre-trained embedding leads us to a concept known as transfer learning.

3.1.2 Transfer Learning

The idea of transfer learning is to first train a general model with a large corpus, long training time, and prominent hardware and then take the model, or its subset, and use it as starting point for other machine learning tasks. In the case of the BERT model, which we use in this study, we take the BERT model - that is, the parameters - and add to it a simple layer for the specific supervised tasks. The BERT model has been trained as a general model that can deal with language in many ways.

So after the pretraining comes tweaking the model for a specific task. That is also known as a fine-tuning phase. For example, we have a labelled data set, such as IMDB reviews, consisting of reviews and positive or negative labels. We add a simple classification layer on top of the pre-trained BERT model. So the input data goes to the pre-trained model as vectors, and at the end, there is only one layer of two nodes: one to indicate a positive review and the other for a negative review. We will finetune the network for this particular task in the learning phase.

The big advantage is that we have to train the pre-trained network with a huge corpus only one time and that training can be done by someone who has access to large computational resources.

So far, we have introduced some ideas on how to represent language in a model. Before we examine the BERT model more carefully, let us briefly introduce some vital concepts to NLP.

Language model

A language model is, in short, a task of predicting what word comes next given a sentence. Formally: Given words $x^{(1)}, x^{(2)}, ..., x^{(t)}$, compute probability distribution for next word $x^{(n+1)}$. In more formally

Input: Sequence of words $x^{(1)}, x^{(2)}, ..., x^{(t)}$
Output: probability distribution of the next word : $P(X^{(t+1)}|x^{(t)}, ..., x^{(1)})$

For NLP, that means that we can pre-train a model, e.g. Wikipedia text, take sentences from it, apply a masking technique of some sort, and use it as training data for the model by exposing sentences one by one. For example, let us look at an example sentence *The weather is good, the sun is shining.*. We then mask some of the words and tell the model to predict the next word. Such as input: *The weather is good, the sun* and give the correct output as *is*. After that, input: *The weather is good, the sun is* , and output: *shining,* etc. In that way, the model learns context representations of the language itself. The BERT model we are using in this study doesn't rely on the classical language model but on a modified version of it. We will discuss more of this later.

3.1.3 Types of neural networks for NLP

So far, we have not talked much about the layers inside neural networks. If you are familiar with computer vision, you may have heard of CNNs, which are building blocks of neural networks used in computer vision. They can detect simple shapes like circles. Alike in NLP, there are similar structures. Let us take a look at some of them.

3.1.4 RNNs and LSTMs

The BERT model does not utilize RNNs. However, they are a crucial component of NLP, and many of the papers refer to them. Hence we will briefly introduce them.

Simple neural networks have no notion of history in them. Let us think about the following sentence and its continuation. *He didn't want to cross the street because it ....* What would be a reasonable follow-up to the sentence? It depends on what the word *it* refers to. The model should somehow know what the word *it* refers to. Recurrent Neural Networks have the ability to have a state that remembers the previous inputs.[6] This is because the network is interconnected in a way that cell reading information from feature t. also can read input from previous RNN cells and hence from all preceding 1...t input features. See 3.3.

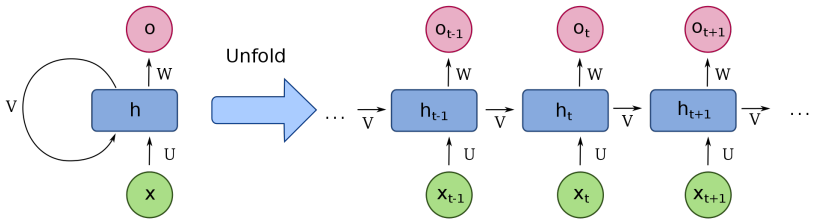

Figure 3.3: RNN unrolled. Source: `https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`, Author: *fdeloche*, Licence: CC4.0

When unrolled, one can see that an RNN is just a long neural network (3.3). However, because each unit is dependent on the output of other previous RNN units, they cannot be parallelized and are thus slow to train. Moreover, the network may not be able to transfer knowledge from far away parts of the network since near-zero gradients which have to go through the network are likely to vanish. Long Short Term Memory LSTM neural network can address the vanishing gradient problem. [7]

Long Short-Term Memory neural units were introduced by Hochreiter and Schmid-

huber in 1997[7]. Unlike the RNN, which has a simple mechanism of passing the output to the second cell h_t, the LSTM has a total of four connections in each cell. The information running through C_{t-1} in the upper part of Figure 3.4, let the information run freely and unaltered. The rest of the cell controls the memory - the state - of an individual cell, making it possible for it to forget and remember information.

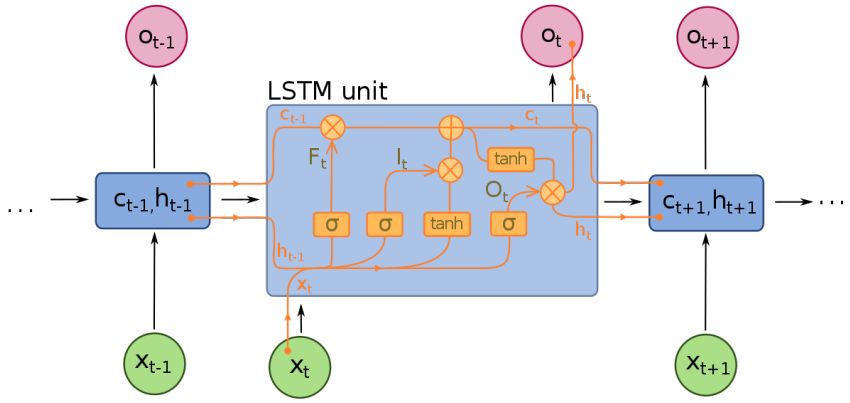

Figure 3.4: LSTM overview. Source: `https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg`, Author: *fdeloche*, Licence: CC4.0

3.1.5 Putting it all together: ELMo and contextual embeddings

Deriving from the notions we have discussed comes the following: Embedding from Language Models (ELMo) (Peters et al. 2018)[8] uses bi-directional LSTMs to do extended word embedding. The problem with traditional word embeddings is that they assign the same word vector to every word despite the fact in some cases, a word may have very different use cases depending on its context. For example, a word mouse can be a little animal but also an electrical device used to control a personal computer. Contextual word embeddings address this problem. ELMo is an example of this.

Now we will start discussing the BERT model that also regards contextual representation, not just word level.

3.1.6 Tokenization

How can an NLP system handle data? As we noted earlier, machine learning systems that deal with images can take an image as an input of pixel values stacked together. Words cannot be directly addressed in this way since there are infinitely many words. Tokenization concentrates on building a vocabulary for an NLP system. If we take all the words present in the training data, we end up with lots of words, and the testing data will likely contain words that were not present in the vocabulary. In this case, a special token, usually called UNK, for the unknown, can be used. But if this is the first time the model sees UNK, the odds are it will affect badly on the performance. One option is to use only TOP M words in the training phase and thus give UNK tokens at words also in the training phase. That, however, leads to the loss of information on those occasions.

The other simple option would be to break all the words word by word and use that feed of individual characters as the input and tokenization system. The downside of that is that the input data get large, and also the structure of the words gets lost as they break into individual atoms. The approach between these two ways is subword tokenization, in which words are broken into subwords. For example, greater can be broken into great-er, lower to low-er. This technique makes it possible to include rare words such as fatelessness into fate-less-ness rather than replacing them without of the vocabulary item UNK.

The BERT model we use in this study uses the subword method, with hash-tags at the beginning or at the end to indicate whether the token is added to a word. For example, a sample sentence Methanoregula formicica sp. nov., a methane - producing archaeon isolated from methanogenic sludges... would be represented as "Met", "##han", "##ore", "##gu", "##la","form", "##ici", "##ca", "s", "##p", ".", "no", "##v", ".", "a", "met", "##hane","-", "producing", "arch", "##ae", "##on", "isolated", "from", "met", "##han", "##ogenic",

"s", "##lu", "##dge", ".". Few thing to note here is that the word Methanoregla is split into five tokens while word *producing* is just as it is: one token. The former is rare even in medical context meanwhile the latter is frequent in the English language, so it is logical there is a token the other thing to note is that spaces are omitted and the if there is no double has ’’ at the ending or at the beginning, you can place a space between the token, when reconstructing the word. The BERT model has its own tokenization system that we will discuss later in this chapter.

3.2 Attention mechanism and the Transformer

BERT utilizes a technique known as self-attention. Before diving into BERT, let us familiarize ourselves with the concept.

Remember the RNNs, we discussed briefly. An RNN has an internal state that it carries on from cell to cell. The weakness of that approach is that especially with long sequences, when information passes from the encoder to the decoder, the state from early cells tends to fade out.

To address this problem Bahdanau et. al.[9] introduced a technique known as attention in 2014. Instead of using fixed-length vectors that pass information from encoder to decoder. Every decoder cell has access to a context vector. See Figure 3.6.

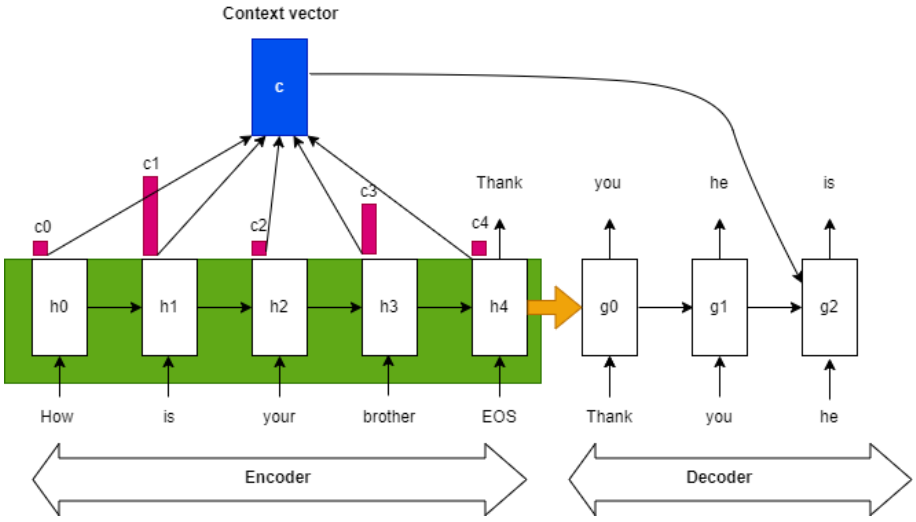

Figure 3.6: Attention mechanism added to RNN encoder-decoder -architecture. For example cell g2 receives both the output from g1 and a context vector c, and from those, it predicts the next word. (Image: Author's own creation, inspired by concepts from 'Learn BERT - the most powerful NLP algorithm by Google' by Martin Jocqueviel)

3.2.1 Attention is All you need and the Transformer

In 2017 Vaswani et al. proposed in a paper *Attention is All you need*, a new model, the Transformer. Instead of RNNs, GRUs or LSTMs, they argued that the model could be built solely on the attention mechanism called self-attention. Let's take a look at some of the building blocks of the transformer model.

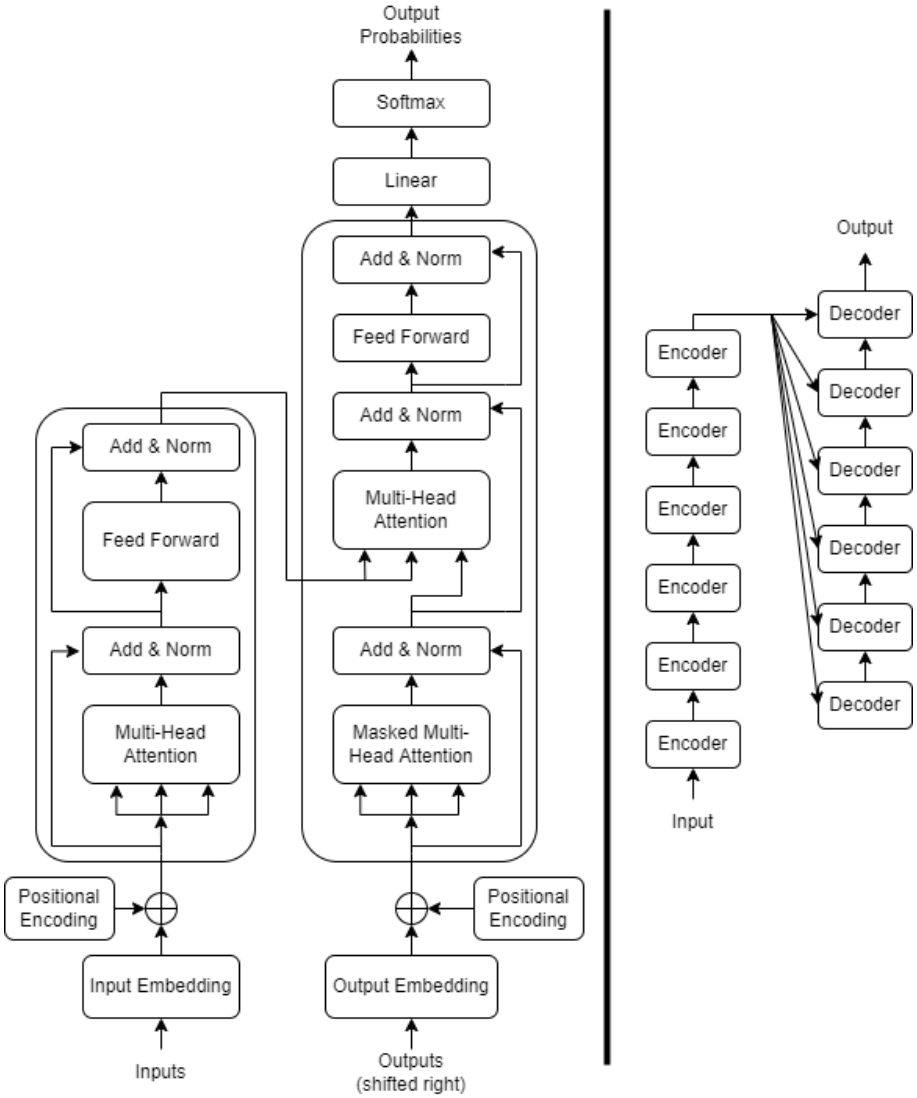

Figure 3.7: The Transformers model architecture. (Image: Author's own creation)

Encoder and Decoder

Both the encoder and the decoder consist of six identical layers, as illustrated on the right side of Figure 3.7. The decoder has two sub-layers: multi-head self-attention and position-wise fully connected feed-forward network. The decoder has a similar structure but has an additional multi-head self-attention layer which gets parts of its input from the encoder (Figure 3.7 left). The first multi-head attention mechanism is slightly modified on the decoder side: A masking strategy is used to hide the subsequent positions from the outputs.

Scaled Dot-Product Attention

The paper introduces a particular type of attention called Scaled Dot-product Attention, which uses concepts of *keys*, *queries* and *values*.

Let us walk through the basic steps of attention:

1. Input matrix x is multiplied by weight matrices W^q, W^k and W^v to obtain respectively Q, K and V matrices(see Figure 3.8). These are representations for the inputs.

2. After obtaining Q, K and V, the matrix Q is multiplied with the transpose of K: K^T and then the product matrix is divided by a scaling factor $\sqrt{d_k}$, where d_k is the dimension of K and Q. Without the scaling factor, the QK^T operation would bring the softmax function to regions where it has very small gradients.

3. Then the softmax function: $\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$) is applied to result. The softmax function basically makes bigger numbers even bigger and smaller even smaller.

4. Finally, the output is multiplied by matrix V. In matrix representation attention is:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V = Z \tag{3.2}$$

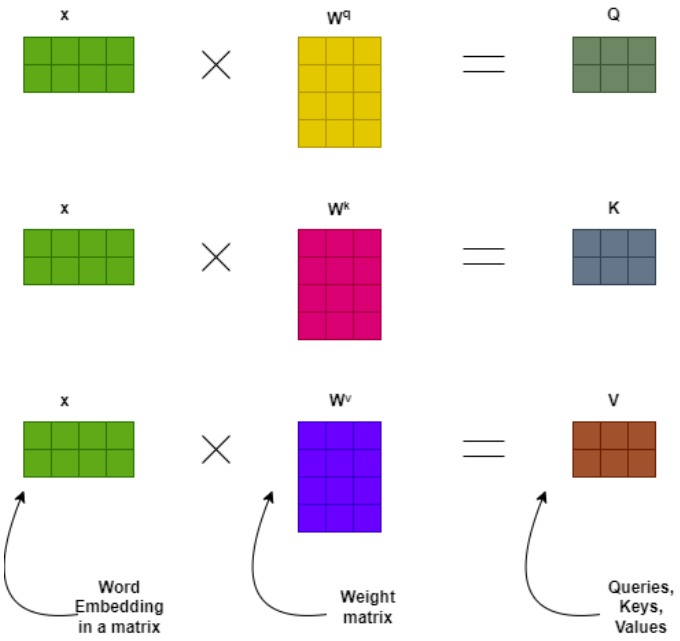

Figure 3.8: Matrices: Q, K and V are obtained from x and by their respective weight matrices W. (Image: Author's own creation)

Multi-head attention

The attention mechanism can be done multiple times simultaneously. That is known as multi-head attention. The great thing in comparison with RNNs is that this can be done in parallel, which will save computational time. In the paper, 8 heads are used in parallel. Compared to the attention procedure we went through, in multi-head attention, instead of one set of Q, W, V matrices, we have a set of them for each attention head Q_i, K_i and V_i. Figure 3.9 depicts the process inside the heads.

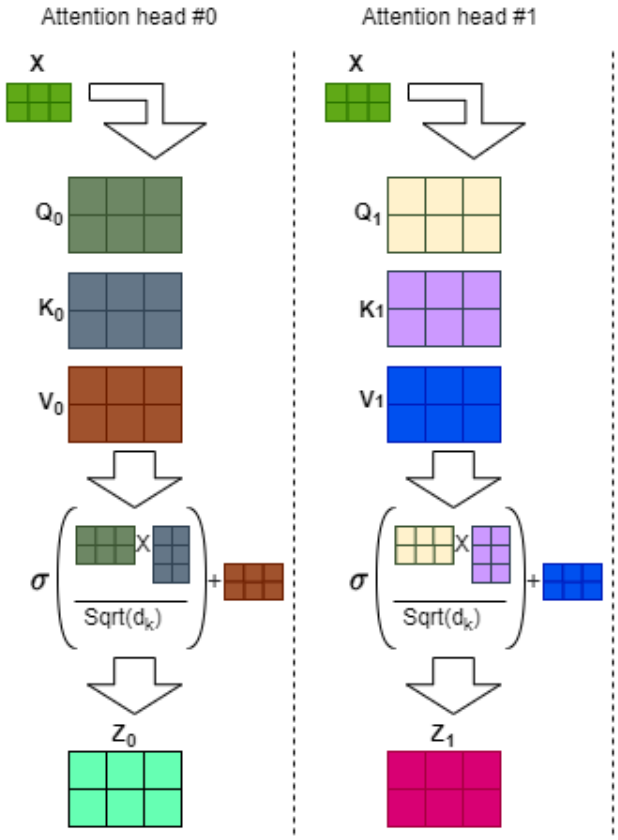

Figure 3.9: Attention heads enable the model to run in parallel. (Image: Author's own creation, inspired from work of Jay Alammar: `https://jalammar.github.io/illustrated-transformer/`)

As each of the attention heads outputs a matrix, these matrices must be compressed together. This is done by first concatenating the result matrices: Z_0,Z_1,...,Z_n and then by multiplying the new long matrix by a weight matrix W to get the final matrix Z (as depicted in Figure 3.10).

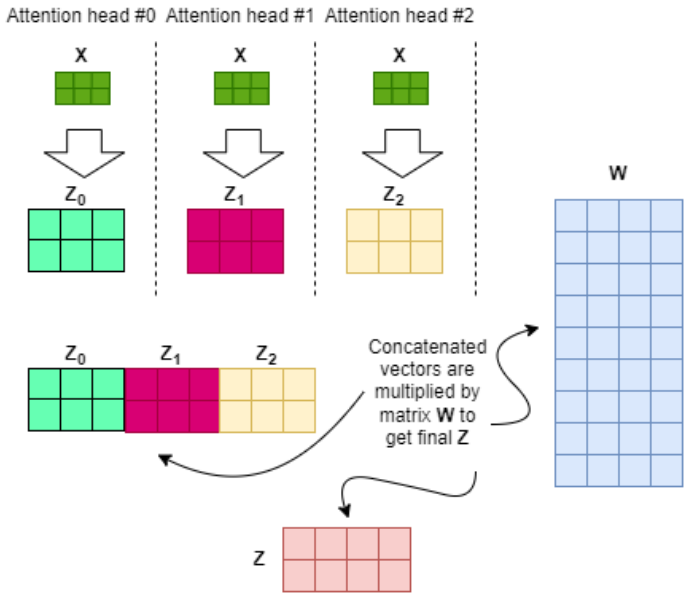

Figure 3.10: The output of the attention heads is concatenated and multiplied by the weight matrix. (Image: Author's own creation, inspired from work of Jay Alammar: `https://jalammar.github.io/illustrated-transformer/`)

The attention mechanism was first used in computer vision to aid in examining which part of the image a model focuses on. Similarly, the same information can be retrieved by looking at the attention heads. Each head will focus on slightly different parts of the input text. Let's look at an example:

Example: Help from the visualization - Attention heads

Let us consider a translation task using an attention-based model. It is possible to examine which part of a sentence is used when translating a word by looking inside the attention matrices.

Let us consider an English sentence and its Spanish translation:

In English	try to Figure it out . $< end >$
In Spanish	$< start >$ trata de averiguarlo . $< end >$

Table 3.1: Translating between Spanish and English.

We can see from the diagram that the translation of each word relates to multiple words in the input. So, for example, word *averiguarlo* uses the words to,figure, it,out in the translation process, and not just a single word. See Figure 3.11.

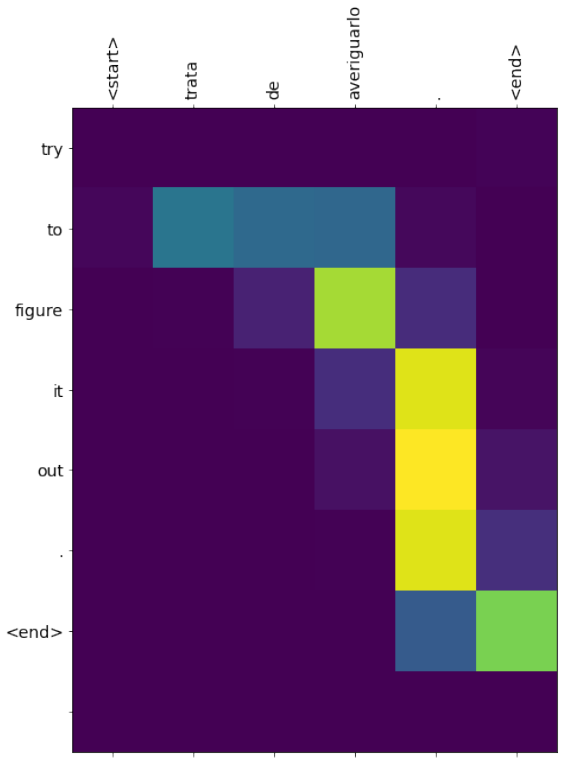

Figure 3.11: Attention mechanism illustrated in spanish-english translation. Source: `https://www.tensorflow.org/tutorials/text/nmt_with_attention`, Author: tensorflow.org, Licence: CC 4.0

Positional encodings

The transformer model has no recurrence nor convolution, and therefore it has no way to know the order of sequence. A special kind of technique can be used to

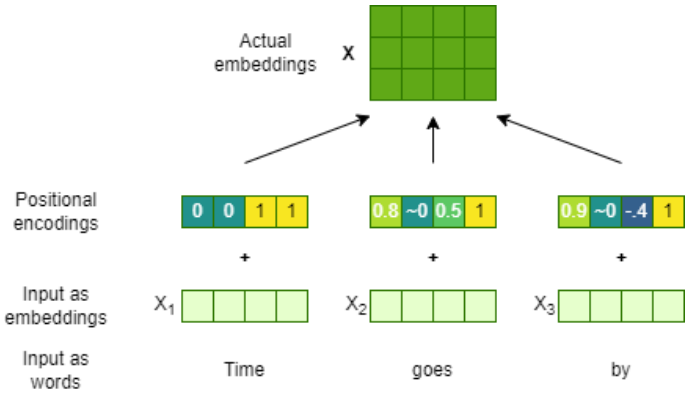

Figure 3.12: Words are turned into embeddings, and the embeddings are summed up with positional encoding vectors to achieve X, which is fed into the model. (Image: Author's own creation)

make chronological order to the model: positional encoding. The transformer uses positional encoding before feeding input to the encoder and decoder. A positional encoding vector is added to each input word embedding. The positional encoding vector has the same dimension as the embedding vector. Positional encodings advance in time, so no two similar vectors can occur. See the Figure 3.12 as an example.

To look more closely at how positional embedding vectors are formulated, let us see the equations in the formula 3.3.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$\mathrm{PE}_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

(3.3)

where:

- *pos* is the position of a word in a sequence.
- i is the i-th dimension of a word vector. For i: $0 \leq i < \frac{d_{model}}{2}$.
- *d model* - denotes the dimension of the output embedding space.

Notice that the terms above are the same, except the first one uses sine and the latter cosine. So, for example, in the case of 64-dimension input, we would have the following positional encoding.

$$new\_embedding_{word@pos} = embedding\_of\_word_{word@pos} + positional\_encoding\_vector$$

which equals to:

$$= e_{word@pos} + [sin(\frac{pos}{10000^0}), cos(\frac{pos}{10000^0}), sin(\frac{pos}{10000^{2/64}}),$$
$$cos(\frac{pos}{10000^{2/64}}), ..., sin(\frac{pos}{10000^{62/64}}), cos(\frac{pos}{10000^{62/64}})]$$

The Figure 3.13 shows how positional encodings for 10 embeddings of size 64. The benefit of using sine and cosine to encode time is that the number of words(tokens) can easily be varied.

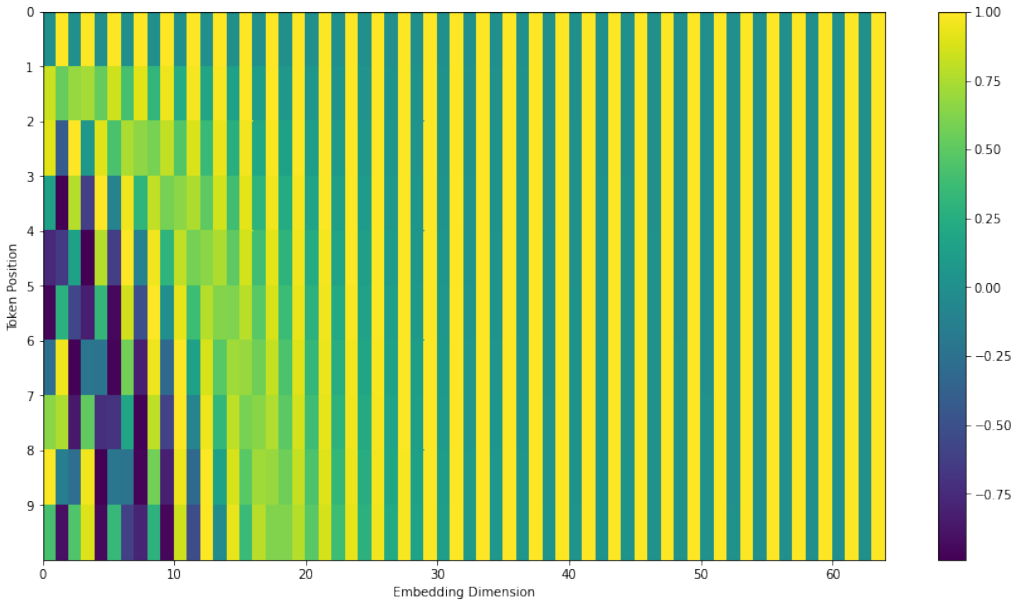

Figure 3.13: Positional encoding. Source: `http://jalammar.github.io/images/t/attention-is-all-you-need-positional-encoding.png`, Author: *Jay Alammar*, Licence: CC4.0

3.3 BERT - Bidirectional Encoder Representations from Transformers

In this study, we use BERT, a general language model from Google AI Language by Devlin et al.[10]. As its name implies, BERT uses the encoder part of the transformer, which we discussed in the previous section. At its release, BERT obtained 11 state-of-the-art results on various NLP tasks. Before the release of BERT, there were two main strategies for using pretraining language representations: feature-based models such as ELMo[8] and fine-tuning-based models such as Generative Pre-trained Transformer (OpenAI GPT)[11]. They are both unidirectional, meaning that every token can only see either previous tokens (left-to-right) or the following tokens (right-to-left). See Figure 3.14. Devlin et al. argue that it is bidirectionality that is the key to the improvement of such models. Hence the abbreviation: Bidirectional Encoder Representations from Transformers.

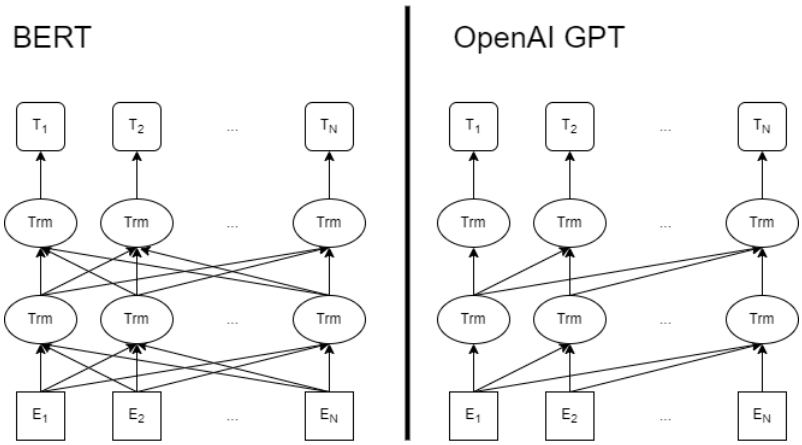

Figure 3.14: BERT is bidirectional, whereas OpenAI GPT is unidirectional. (Image: Author's own creation)

As discussed earlier, a traditional unsupervised language model is trained by masking the next word and making a model predict it. There is also an older variant of this technique, the Masked Language Model (MLM), where some randomly chosen tokens are masked, and a model is trained by making it guess the right word for the

masked tokens. This technique is also known as the Cloze task[12].

There are two steps involved in the BERT framework: pretraining and fine-tuning. As stated earlier: *Pre-trained word embeddings offer a significant improvement over embeddings that are learned from scratch*[5]. The model is trained by vast amounts of unlabeled data in the pretraining phase. After the training process, the model can be used for various specialized problems: the pre-trained model is *fine-tuned* on specific downstream-task. Downstream is a term the deep learning community uses to refer to attaching (a simple) neural network or alike at the end of a pre-trained model. In this way, the pre-trained model can be trained only once, which requires a lot of work and can be fine-tuned to excel on specific tasks. In the case of BERT, there is only a tiny difference between the pre-trained model and the downstream architecture[10].

3.3.1 Architecture

BERT is based on the transformer[13]. Unlike the complete transformer we discussed earlier, BERT uses only the encoding part of the transformer. BERT comes in two different model settings. The base model has 12 transformer blocks and 110 million parameters where as large model has 24 blocks and 340 million parameters. Models are more detailed in Table 3.2.

Type	Transformer blocks	Hidden size	Self-attention heads	Parameters in total
base	12	768	12	110 M
large	24	1024	16	340 M

Table 3.2: Two different BERT models: Base and large

3.3.2 Pretraining process

Pretraining of BERT is carried out in a way that handles two tasks at the same time: Making the model predict whether two sentences logically follow each other and guessing the right word for masked tokens.

Each input begins with a special [CLS] token. In the downstream tasks, the corresponding output token of the [CLS] token can be used for, e.g. classification. There is also another special token known as separator [SEP], which is used to separate two different sentences A and B. An additional embedding is added to the embeddings of the words to indicate whether a word belongs in to sentence A or sentence B.

Masking entities

Language models are often trained by exposing an input one by one from left to right and making the model predict the next word. This work just fine, for example, for OpenAI GPT-model, whose intrinsic nature is from left to right. However, as BERT is designed to be able to use the whole context, this strategy does not apply. There is, however, another similar method known as Cloze[12]. The learning is done by selecting tokens at random and replacing them with special [MASK] tokens. In the pretraining of the BERT model, 15 % of the words were masked.

However, as there is no [MASK] identifier in downstream tasks in general, a more complex strategy is being used. See the Table 3.3.

80 %	replace with [MASK]	my dog is [MASK]
10 %	replace with a random word	my dog is apple
10 %	keep the word unchanged	my dog is hairy

Table 3.3: Masking the word hairy in *my dog is hairy*

The idea behind this is that it the Transformer encoder to keep a distributional

contextual representation of every input token because the Transformer encoder does not know which words will be asked to predict and which have been replaced by a random word[10].

Next sentence prediction

The BERT model is trained to differentiate a reasonable consecutiveness of two sentences to make the model fit for downstream tasks such as Question Answering. The first part of the input, from [CLS] to [SEP], is a sentence A and the other from [SEP] to the end is a sentence B. The training data is produced in such a way that in 50 % of the cases, sentences A and B are subsequent, and in 50 % of the cases, they are random: e.g. sentence A: *To be or,* and sentence B: *not to be.* versus A being *To be or,* and B being *The black cat crossing the street.* These are labelled *IsNext* and *NotNext.*

[CLS] the man went to [MASK] store [SEP] he boughta gallon [MASK] milk [SEP]	IsNext
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight less birds [SEP]	NotNext

Table 3.4: Example sentence inputs[10].

Evaluation architecture choices

BERT has two distinct architectural characteristics: bidirectionality and next sentence prediction. BERT paper tries to evaluate how much impact they have on the model performance. The Table 3.5 shows the comparison. The following interpretation can be deduced:

- **No NSP** - Leaving the next sentence prediction off hurts SQuAD, QNLI and MNLI tasks.

- **LTR & No NSP** - left-to-right language model with no next sentence prediction but rather masked language model: Performance is worse on all of the tasks.

- **LTR & No NSP** and + **BiLSTM** a LTR model will perform poorly at token predictions since the token-level hidden states have no right-side context.

To make the results more comparable BiLSTM adds a randomly initialized BiLSTM on top of the while it makes results better, and they are still behind the bidirectional model.

So, all in all, it seems that the BERT architecture choices are well-grounded.

Tasks	MNLI-m (Acc)	QNLI (Acc)	MRPC	SST-2 (Acc)	SQuAD (F1)
BERTbase	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 3.5: Measuring the effect of architectural choices on the performance of the model[10].

3.3.3 BERT's performance

Devlin et al. performed several tests on BERT and compared the results for those of PreOpenAI SOTA, ELMo and OpenAI GPT. GLUE (The General Language Understanding Evaluation) is a collection of diverse natural language understanding tasks. They involve tasks such as:

- MNLI - Multi-Genre Natural Language Inference is an entailment classification task: Given a pair of sentences A and B. From a set of *entailment*, *contradiction* or *neutral* what is A's relation to B.

- QQP - Quora Question Pairs. Binary classification task: Whether two questions asked on Quora platform are semantically equivalent.

- QNLI - Question Natural Language Inference: A task in which the model tries to know if a sentence holds correct answer for a question.

Apart from these the GLUE also has SST-2, CoLA, STS-B, MRPC and RTE tasks, which are covered here.

As we can see from the Table 3.6. Both BERTbase and BERTlarge excel other models significantly.

System	MNLI-(m/mm)	QQP	QNLI	...	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	...	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	...	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	...	75.1
BERTbase	84.6/83.4	71.2	90.5	...	79.6
BERTlarge	86.7/85.9	72.1	92.7	...	82.1

Table 3.6: Comparing BERT to other model on GLUE task

3.3.4 BERT on Named Entity Recognition: Feature-based approach vs. Fine-tuning

The topic of this thesis is Named Entity Recognition, NER. The BERT paper tested two different approaches to downstream NER tasks: fine-tuning and feature-based method. Feature-based methods, where features are extracted from the pre-trained model to be used as a starting point for another model, are used to use the non-fine-tuned contextual embeddings as input for the two-layer 768-dimensional BiLSTM model. The BERT paper reports the results listed in 3.7 for the CoNLL-203 NER task introduced in 2003[14].

This thesis uses BERT's Large and Base models with the fine-tuning approach,

System	**Dev F1**	**Test F1**
For reference		
Elmo (Peters et al. 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	93.1
BERT-based models		
fine-tuned		
LARGE	96.6	92.8
BASE	96.4	92.4
Feature-based		
Weighted Sum Last Four Hidden	95.9	
Concat Last Four Hidden	96.1	
Weighted Sum All 12 Layers	95.5	

Table 3.7: BERT based methods applied to the CoNLL-2003 task

but an interesting option for further studies would be to try the feature-based methods on the S800 dataset. For fine-tuning the paper gives the hyperparameters listed in Table 3.8 that work well on across all tasks. We used a grid of these parameters in our work.

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

Table 3.8: Example sentence inputs from the BERT paper

3.4 Named Entity Recognition

Named Entity Recognition is a task in information extraction that tries to retrieve and classify named entities in a corpus into categories such as organizations, locations, person names, etc. In the context of medical text, those categories are, e.g. Species, Strains, Genus, Chemical, Gene Product etc.

Our goal is to locate all the species mentioned in the corpus in this work. For this, we first convert the original corpus into standoff format, which specifies the place in the text where the tag in question is present. Then we turn standoff-format into CoNLL-format, which has each word token separated in its own line followed by its classification tag.

3.4.1 Example: A document from the S800

Let's look at an example: the document 19667393 in standoff format of the S800 original data sample:

Methanoregula formicica sp. nov., a methane-producing archaeon isolated from methanogenic sludge.

A novel methane-producing archaeon, strain SMSP(T), was isolated from an anaerobic, propionate-degrading enrichment culture that was originally obtained from granular sludge in a mesophilic up-flow anaerobic sludge blanket (UASB) reactor used to treat a beer brewery effluent. Cells were non-motile, blunt-ended, straight rods, 1.0-2.6 mum long by 0.5 mum wide; cells were sometimes up to 7 mum long. ...

Standoff - format

In the standoff format, there is a separate .ann file that consists of places in the text, such as 0 23, species name Methanoregula formicia and Taxonomy code 882104 (see Table 3.9). The text itself is in the corresponding .txt file.

T1	Species 0 23 Methanoregula formicica
N1	Reference T1 Taxonomy:882104
T2	Species 143 150 SMSP(T)
N2	Reference T2 Taxonomy:882104
T3	Species 1076 1083 SMSP(T)
N3	Reference T3 Taxonomy:882104
T4	Species 1225 1249 Methanoregula boonei 6A8
N4	Reference T4 Taxonomy:358766
T5	Species 1423 1430 SMSP(T)
N5	Reference T5 Taxonomy:882104
T6	Species 1435 1459 Methanoregula boonei 6A8
N6	Reference T6 Taxonomy:358766
T7	Species 1770 1777 SMSP(T)
N7	Reference T7 Taxonomy:882104
T8	Species 1863 1886 Methanoregula formicica
N8	Reference T8 Taxonomy:882104
T9	Species 1918 1925 SMSP(T)
N9	Reference T9 Taxonomy:882104

Table 3.9: Standoff format

CoNLL - format

The same data turned into CoNLL-format is listed in Table 3.10. The text data is separated on each line, followed by its NER tag. In this example, *B-Species* means the Species tag that begins, and *I-Species* Species tag continues. *O* tag means that word is not a named entity.

Word	**Tag**
Methanoregula	B-Species
formicica	I-Species
sp	O
.	O
nov	O
.	O
,	O
a	O
methane	O
-	O
producing	O
archaeon	O
isolated	O
from	O
methanogenic	O
sludge	O
.	O
...	...

Table 3.10: CoNLL format

3.5 NER with BERT

3.5.1 BioBERT model

Each BERT model also has a tokenization file *vocab.txt*, which contains the tokens that are used to split each word. The first token [PAD] is used to fill up the 512-window so that each input element has a token. [UNK] token is used for unknown words, [SEP] is used to separate the two sentences that either follow or are randomly selected in the pretraining process. A token that begins with double hash ## such as 1466 ##ia is used in splitting rare words into tokens. Here are some samples from the *vocab.txt* file, as shown in Table 3.11.

Table 3.11: Biobert model has 28996 different tokens by which any token is split.

Line number in vocab	**Token**	**Line number in vocab**	**Token**
1	[PAD]	2	[unused1]
3	[unused2]	...	...
100	[unused99]	101	[UNK]
102	[CLS]	103	[SEP]
104	[MASK]	105	[unused100]
106	[unused101]	107	!
108	"	109	#
...	...	...	...
1465	felt	1466	##ia
1467	station	...	...
3813	scientific	3814	sales
3815	##ai	3816	theme
3817	starts	3818	clearly

Line number in vocab	**Token**	**Line number in vocab**	**Token**
3819	##ut	...	...
28994	##-	28995	##/
28996	##:		

Example from S800 dataset

The first sentence in the training data is *Methanoregula formicica sp. nov., a methane-producing archaeon isolated from methanogenic sludge.* Tokenization splits each word into tokens that are in the vocabulary. For example *Methanoregula* becomes *Met, ##han, ##ore, ##gu, ##la.* The input window size in BERT is 512 tokens. The first token is [CLS], and the tokens of the sentence follow it. If there is space left after the sentence, the rest is filled with [PAD] token all the way to the 512.

1 - [CLS]	9 - ##ca	17 - a	25 - isolated	33 - .
2 - Met	10 - s	18 - met	26 - from	34 - [PAD]
3 - ##han	11 - ##p	19 - ##hane	27 - met	35 - [PAD]
4 - ##ore	12 - .	20 - -	28 - ##han	36 - [PAD]
5 - ##gu	13 - no	21 - producing	29 - ##ogenic	...
6 - ##la	14 - ##v	22 - arch	30 - s	511 - [PAD]
7 - form	15 - .	23 - ##ae	31 - ##lu	512 - [PAD]
8 - ##ici	16 - ,	24 - ##on	32 - ##dge	

3.5.2 Cross-sentence method for NER

The traditional way to make a Named Entity Recognition task is to feed the model one sentence at a time and predict the outputs. However, as BERT's attention mechanism can access information from long distances in the input data, it would

seem intuitively logical that the model could perform better when it sees context information around the target. Jouni Luoma and Sampo Pyysalo have studied this in the paper *Exploring Cross-sentence Contexts for Named Entity Recognition with BERT*[15]. We don't use the CMV method described in the paper fully, but a light version of it is discussed in the paper that takes into account the subsequent sentences of the target sentence.

Example of S800 sentence in Cross-sentence representation

Let us look at a small example of how the sentences are represented in the data model fed to BERT. Here are a few sample sentences from the S800 dataset.

- **Sentence 1:** *Methanoregula formicica sp. nov. a methane-producing archaeon isolated from methanogenic sludge.*
- **Sentence 2:** A nmethane-producing archaeon, strain SMSP (T), was isolated from an anaerobic, propionate-degrading enrichment culture that was originally obtained from granular sludge in a mesophilic upflow anaerobic sludge blanket (UASB) reactor used to treat a beer brewery effluent.

In this case the BERT input-window would be the following, after it has been tokenized:

1 - [CLS]	8 - ##ici	15 - .	22 - arch	29 - ##ogenic
2 - Met	9 - ##ca	16 - ,	23 - ##ae	30 - s
3 - ##han	10 - s	17 - a	24 - ##on	31 - ##lu
4 - ##ore	11 - ##p	18 - met	25 - isolated	32 - ##dge
5 - ##gu	12 - .	19 - ##hane	26 - from	33 - .
6 - ##la	13 - no	20 - -	27 - met	34 - [SEP]
7 - form	14 - ##v	21 - producing	28 - ##han	35 - A

36 - novel	54 - from	71 - that	89 - ##flow	107 - a
37 - met	55 - an	72 - was	90 - an	108 - beer
38 - ##hane	56 - an	73 - originally	91 - ##ae	109 - brewery
39 - -	57 - ##ae	74 - obtained	92 - ##ro	110 - e
40 - producing	58 - ##ro	75 - from	93 - ##bic	111 - ##ff
41 - arch	59 - ##bic	76 - g	94 - s	112 - ##lue
42 - ##ae	60 - ,	77 - ##ran	95 - ##lu	113 - ##nt
43 - ##on	61 - prop	78 - ##ular	96 - ##dge	114 - .
44 - ,	62 - ##ion	79 - s	97 - blanket	115 - [SEP]
45 - strain	63 - ##ate	80 - ##lu	98 - (	SENTENCE 3
46 - SMS	64 - -	81 - ##dge	99 - U	[SEP]
47 - ##P	65 - de	82 - in	100 - ##AS	SENTENCE 4
48 - (	66 - ##grad-	83 - a	101 - ##B	. . .
49 - T	ing	84 - me	102 -)	SENTENCE N
50 -)	67 - en	85 - ##so	103 - reactor	. . .
51 - ,	68 - ##rich	86 - ##phi	104 - used	511 - [PAD]
52 - was	69 - ##ment	87 - ##lic	105 - to	512 - [PAD]
53 - isolated	70 - culture	88 - up	106 - treat	

And the corresponding labels in the output are marked as.

1 - O	8 - I-Species	15 - O	22 - O	29 - O
2 - B-Species	9 - I-Species	16 - O	23 - O	30 - O
3 - I-Species	10 - O	17 - O	24 - O	31 - O
4 - I-Species	11 - O	18 - O	25 - O	32 - O
5 - I-Species	12 - O	19 - O	26 - O	33 - O
6 - I-Species	13 - O	20 - O	27 - O	34 - [SEP]
7 - I-Species	14 - O	21 - O	28 - O	35 - O

36 - O	43 - O	50 - I-Species	504 - [PAD]	511 - [PAD]
37 - O	44 - O	51 - O	505 - [PAD]	512 - [PAD]
38 - O	45 - O	52 - O	506 - [PAD]	
39 - O	46 - B-Species	. . .	507 - [PAD]	
40 - O	47 - I-Species	501 - [PAD]	508 - [PAD]	
41 - O	48 - I-Species	502 - [PAD]	509 - [PAD]	
42 - O	49 - I-Species	503 - [PAD]	510 - [PAD]	

The test set data used to evaluate the model is treated the same way. Sentences and labels are separated by [SEP] tag, concatenated and fit into the 512-window. However, in the last step, before evaluating the predicted labels against the real labels, additional sentences are chopped off in a way that there is the only one prediction for every sentence.

3.6 Other NER-methods

So far we have discussed only neural network based approaches for Named Entity Recognition. This sort of approach is quite novel and there are many other more traditional and human understandable methods for NER problems. They are better summarised in the LINNAEUS: A species name identification system for biomedical literature[16]. We will briefly discusse them here. At large, there are two types of system for NER: (1) rule based systems and (2) dictionary-based systems.

3.6.1 Rule based-systems

Generally organism names occur in sequences called *binomen* and *trinomen* and are derived from either Latin or Greek. First genus part is capitalized and preceding names are written in lowercase. This systematic naming convention makes it possible to build rule based tagging systems to extract species names. Two well know taggers that build upon this idea are called TaxonGrab[17] and FAT[18]. TaxonGrab aids the process of identifying species names also with a dictionary based method. As the scientific names comes mainly from Latin or Greek, filters can be used to separate taxonomic name candidates using language-specific lexicon.

TaxonGrab paper states that their system performs at over 96 % at both precision and recall[17]. However their evaluation corpus was a single book *The Birds of the Belgian Congo* by James Paul Chapin, which consists of 5000 pages and over 8000 taxonomic names. As this book is by one writer/editor, it is likely that the naming conventions are quite concise and as the name suggests they are from a quite narrow field. However F-score over 96 % is quite impressive and shows that deterministic models can perform very well.

Such rule based methods are easy to debug as they build upon rules and their inner workings are human-friendly but they surely have limitations, as there are lots of common names, and the naming conventions might not always be very concise in

all biomedical articles.

Rule based models can also be used to identify names, that have not been recorded in existing species dictionaries[16]. According to Gerner rule-based methods cannot often identify common names.

3.6.2 Dictionary based systems

Dictionary based methods build upon a dictionary to identify names. They have also one crucial benefit compared to other systems: they can map species names to unique database identifiers. That all comes with cost, they require a regularly updated dictionary of new names. In this category the LINNAEUS has been widely used[19]. Gerner et al. argue that common term such as *human* are in general hard for rule-based systems and that dictionary-based methods are better at finding common names and hence more suitable for species recognition in biomedical literature[16]. We will take closer look at the the LINNAEUS system.

Overview of *the LINNAEUS: A species name identification system for biomedical literature*

As the LINNAEUS paper states there are several challenges in identifying species names: (These are full quotations from the LINNAEUS paper)[16].

1. *Species name ambiguity: many abbreviated species names are highly ambiguous (e.g. 'C. elegans' is a valid abbreviation for 41 different species in the NCBI taonomy.)*

2. *Homonymy with common words: some species common names are widely used in generak English text (e.g. 'Spot' for Leiostomus xantherus*

3. *Acronym ambiguity: species dictionaries contain acronyms for species names (e.g. HIV for Human immunodeficiency virus), which can refer to multiple*

species

4. *Variability: while species dictionaries cover a large number of scientific names, synonyms and even some common misspellings, they cannot match human author in variability of term usage*

Here we have summarized some special features of the LINNAEUS system

- **Abbreviation generation**: New abbreviations are derived from longer ones. E.g from *Drosophila melanogaster* we can get *D. melanogaster*. After this the final dictionary has on average 2.46 names for each individual original entity.

- **Common synonyms addition**: A set of additional synonyms were included which occur frequently in the literature. E.g *patient* is such a term, that is assumed to refer *human*.

- **Regular expression optimization**: As there are many terms in the dictionary (NCBI taxonomy has 386 108 Species in June 2009), assigning a regular expression to each entity would require lot of time resulting in a slow model. However with the help of deterministic finite-state automatons (DFA) allow the regular expression to be combined and the algorithm to run in $O(n)$ time, where n being the length of the text.

- **Longest match principle**: When there are several species names that overlap but differ in length, the longest one is selected. Hence *Human immunodefiancy virus*, will match the longest form instead of just *Human*.

- **Context disambiguation**: In cases when abbreviated mentions that refer to many different species, and one of the candidates appear in the text, all the abbreviated mentions are set to point to that particular species. E.g *C. elegans* can point to 41 different species, but suppose there appears also

Caenorhabditis elegans in a text, in that case all the *C.elegans* abbreviations will point to *Caenorhabditis elegans*.

- **Removal of common words**: Some species names are also common words in the English language, such as spot for *Leiostomus xanthurus*. These kind of words are removed from the species dictionary to prevent appearance of large number of FNs.

- **Probability assignation**: A probability distribution is assigned to mentions that do not have a specific single species identifier that they could be set to point to. The probabilities are obtained by Acromine[20]. In this way unspecific mentions of HIV, will have a more probability mass for *Human immunodeficiency virus* than for *the Hippocratic irrelevance variable*, for which the term is also used.

The LINNAEUS paper uses term *Document level performance* and *Species level performance*. In the scope of document level performance term it is enough to find out if a species exists in a certain document or not and in species level the exact position must also be found. The LINNAEUS system is tested against several different data sets, such as *WhatizitOrganisms* and *Entrez*. Precision and recall performances vary from 10 % to 99 % in these ealuation data sets and in some cases, such as with Entrez ambiguous terms such as *Drosophila Melanocaster* accounting for a third of FN. (Same term causing headache also on our study). For a manually annotated corpus the LINNAEUS perform 94.3 % recall and 97.1 % precision on mention level and 98.1 % recall and 90.4 % precision on a document level.

Overview of *The SPECIES and ORGANISMS Resources for Fast and Accurate Identification of Taxonomic Names in Text*

In this subsection we have summarized the SPECIES and ORGANISM paper[19]. The SPECIES model has similar characteristics as the LINNAEUS system. We will list here some key features:

- The SPECIES tagger is based on NCBI Taonomy dictionary. Names higher (genus and higher) and lower (strain and lower) of species.
- **Automatic abbreviations**: From full the LINNAEUS name abbreviated form were automatically made(*Cannabis sativa* -> *C. sativa*)
- **Inclusion of common names**: A small number of common names was added to the dictionary
- **Orthographic variation**: Species names can be written as one word, two word or with hyphen. To standardize this variation custom-made hashing and string-compare functions are applied.
- **Longest match**: Documents are tokenized based on whitespaces and some special tokens. Substrings consisting six tokens are used against the dictionary to make sure the longest match is taken.
- **Acronym normalization**: If acronyms are followed by a species name in parenthesis, they are mapped to that particular species. If there are no parenthesis, other suitable candidates are looked from the document. E.g *C. sativa* can refer to *Camelina sativa*, *Cannabis sativa* or *Castanea sativa*. If one of these full names are found in the document, all *C. sativa* mentions will be normalized to it.
- **Exclusion of common words causing FPs**: Some words cause of a lot of

FPs. These can be found running the tagger on Medline abstracts and from word frequencies these hard words can be determined.

As we can see, SPECIES and the LINNAEUS taggers share similar characteristics and their evaluation performances are roughly the same. The main difference between these taggers comes with speed, SPECIES being a magnitude faster and more memory efficient than the LINNAEUS. The speed can be explained by expanded dictionary to hash table, which allows fast lookup of names. The hash table utilizes custom hash and string comparison functions to deal with orthographic variations (to make words such as zebrafish, zebra-fish and zebra fish to be equivalent from the perspective of the tagger).

Table 3.12 shows summary of evaluation performance of the two models tested against the manual annotated corpora S800 and L100E (LINNAEUS). Mention level requiring strict matching where as document level requiring only information whether an entity is present in a document or not. We can see that the two taggers are near each other in evaluation performance. The data set L100E was used in building SPECIES, so it is obvious SPECIES performs better with that particular data set. We can see that the average over all listed F1-scores in the Table for the LINNAEUS is 86 % and for SPECIES 87 %.

corpora	level	software	precision	recall	F1
S800	Document	LINNAEUS	86.4 %	89.3 %	87.9 %
		SPECIES	85.9 %	89.8 %	87.8 %
	Mention	LINNAEUS	84.3 %	75.4 %	79.6 %
		SPECIES	83.9 %	72.6 %	77.8 %
L100E	Document	LINNAEUS	89.2 %	91.4 %	90.3 %
		SPECIES	89.9 %	94.3 %	92.0 %
	Mention	LINNAEUS	88.7 %	81.8 %	85.1 %
		SPECIES	91.5 %	90.8 %	91.1 %

Table 3.12: Comparison of the LINNAEUS and the SPECIES taggers. Document level meaning that to match it suffices to find whether an entity is present in the document where as in mention level the exact position must also be determined. The LINNAEUS corpus was used in building of SPECIES tagger, hence the scores are higher when the SPECIES are evaluated against the LINNAEUS[19]

4 Data

4.1 Motivation

The BERT-based approach on NER gives good results on various medical datasets such as the GELLUS for Cell line circa 93 %, the BC2GM for Genes circa 90 %, the CHEMDNER for Chemicals circa 91 %, the LINNAEUS for species circa 87 %. However, the S800 dataset for Species is much poorer, reaching only up to 73 %. Moreover, cross combining corpura: training on the S800 and testing on the LINNAEUS, gave an F-score roughly circa 80 %. Therefore it raised an interest to analyze the the S800 further. It soon became evident that the LINNAEUS and the S800 data sets were not consistent with each other with respect to annotation rules. This difference is most evidently present with the word *patient*, which is the most frequent word in the LINNAEUS was not tagged at all in the S800.

4.2 The two data sets

4.2.1 S800

The S800 is originally a data set for validating the SPECIES tagger. It has been discussed more thoroughly at the end of the previous chapter. It is a novel abstract-based manually annotated corpus. The S800 comprises 800 PubMed abstracts in which organism mentions were identified and mapped to the corresponding NCBI

Taxonomy identifiers. The S800 abstracts were collected by selecting 100 abstracts from the following 8 categories: bacteriology, botany, entomology, medicine, mycology, protistology, virology and zoology. The S800 has been annotated to address species-level recognition. However, higher taxa mentions (such as genera, families and orders) are also added.

4.2.2 The LINNAEUS

As discussed at the end of the previous chapter the LINNAEUS data set, is a manually annotated data set for the LINNAEUS tagger, and was made originally to evaluate the tagger. But it can be also used for training machine learning models in Named Entity Recognition. According to the authors: *the LINNAEUS is an open-source, stand-alone software system capable of recognizing and normalizing species name mentions with speed and accuracy, and can therefore be integrated into a range of bioinformatics and text-mining applications. The software and manually annotated corpus can be downloaded freely at http://linnaeus.sourceforge.net/*[16]. The species terms were manually annotated and normalized to the NCBI taxonomy IDs of the intended species. *In the cases where a species ID did not exist in the NCBI taxonomy (mostly occurring for specific species strains), they were given a species ID of 0 (which is not used in the NCBI taxonomy)*[16].

4.3 Comparing the two data sets: LINNAEUS and S800-orig

How can we tell something about the quality of annotated corpus? Of course, there are straightforward methods of building a model with the data set and judging the data set by its F-score. That however does not lead us very far. The data set can still be bad, even if it provides good results. Imagine a data set used for species

recognition, in which just a few common names, e.g dog, are tagged. The model could learn easily to recognize dogs as a species and give good results. Its shabby nature would only become visible when tested in real-life environments or tested against another data set designed for a similar task.

In this section, we try to find the similarities and dissimilarities between the two data sets. We have to note that these simple statistical approaches on word-level frequencies might not foretell which model is better as we are using BERT, which looks at the context, not just words. However, the quality has proven to be crucial in many cases and especially it might be here, where there is no data abundance.

4.3.1 Comparing word-level statistics

As the name of the corpus implies, the S800 data set constitutes 800 annotated short abstracts. The LINNAEUS on the other hand is made up of 100 full papers. In the Tables 4.1 and 4.2 we can see that even though the LINNAEUS has around 1.5 times more text in it, the number of annotated species is roughly the same meaning in the S800 the species mentioned are more frequent.

File	**Number Of lines**	**Of which Species**	**% Species**
devel.tsv	23047	869	3.8
test.tsv	43928	1810	4.1
train.tsv	153024	5867	3.8

Table 4.1: File statistics of the S800 original corpus

In Tables 4.3 and 4.4 we can see that common terms dominate the TOP listings in both datasets and in the case of the LINNAEUS terms such as 'patients', 'people', 'women', which are left out in the S800, are very frequent.

File	Number Of lines	Of which Species	% Species
devel.tsv	97648	1064	1.1
test.tsv	171378	2232	1.3
train.tsv	292169	3259	1.1

Table 4.2: File statistics of the LINNAEUS corpus

Number of tags	Species
437	patients
331	human
267	mice
174	mouse
159	patient
112	people
111	women
96	yeast

Table 4.3: TOP10 most frequent words in the LINNAEUS corpus

Number of tags	**Species**
162	human
96	HIV - 1
78	Escherichia coli
72	HIV
68	yeast
64	rice
58	mice
54	HCV
45	Aspergillus nidulans
42	tomato

Table 4.4: TOP10 most frequent words in the S800 corpus

4.3.2 Controversial entities

A statistical approach for this question could be to study what is the density of species tags in a corpus and then compare it to another similar data set. Still, we would need to do a little bit of extra study to find out how often species entities are present on average in the medical literature. Instead, we can try to find inconsistencies within a corpus. In many cases, corpus annotation is done by diving the corpus into smaller subparts and giving them to different people, who can do a little bit of the annotation and then sum it all up. There lies a problem because, although there are guidelines for what should be annotated, people tend to view these guidelines differently.

A quantitative approach

We can examine if there are entities, which are controversial in both data sets. We simply go over the annotation files of a data set and for a list of names of species, which have been annotated. Then we go over the whole corpus and look for entities, which are not annotated. This method has some problems, that we shall discuss shortly. This is a great way of studying some evident cases, but it is still better if we can do this also with another data set. Let's study first the LINNAEUS data set in this way.

For the LINNAEUS we can see, that the inconsistencies are not very abundant. From the fifth word onward there is only 1 In most cases there is only one in the O category. We can see that there are only 20 entities which have hits in both Species and Os.

Here is a short Table 4.5 of entities that are marked both as Species and O in the LINNAEUS. How could we tell which one of these is more inconsistent than another 'woman' is tagged in both categories, but it is tagged 30 times as species and only 1 time as Os. On the other hand, 'bacteriophage' is tagged half of the time as Species

and half as O. But it occurs only 2 times in the corpus. On the other hand, 'rodent' is 3 times as Species and 3 as O. Surely it has more impact than 'bacteriophage'.

	$\text{LIN}_{Species}$	LIN_{Os}
Human	13	1
rodent	3	3
children	53	2
bacteriophage	1	1
Murine	2	2
cyanobacterial	2	3
persons	4	1
women	30	1

Table 4.5: Entities which have both Species and O tags in the LINNAEUS.

From these observations, we shall consider two principles. An inconsistency impact of a word is proportional to how close the number of species is to the number of Os and how many entities are present in a corpus altogether. Let us define:

$$f_{O_impact} = Ratio_O * Presence_{SandO} \tag{4.1}$$

For ratio, a naive way would be to calculate $\frac{O}{S+O}$. However, if O is close to 1, e.g most of the entities are Os and only a few Species, we would like not to penalize a lot because there is not that much inconsistency. Rather we use the following formula:

$$Ratio(S, O) = 1 - 2|\frac{O}{S+O} - 0.5| \tag{4.2}$$

This function has a neat property of penalizing most entities, which have #Species close to # O, and giving a little penalty to case when #Species $\ll$ #O or #Species $\gg$ #O. The Figure 4.1 shows a plot of this function.

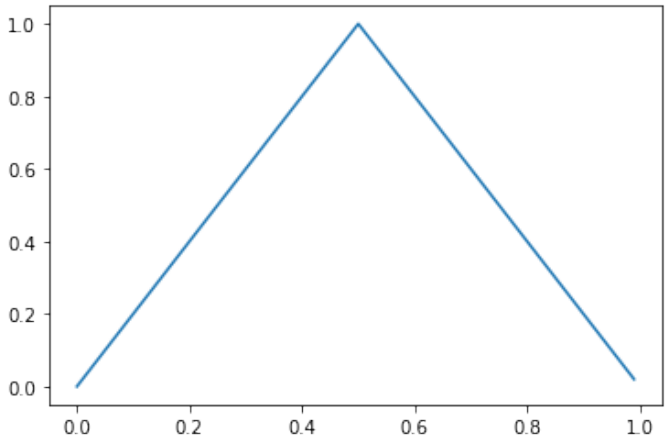

Figure 4.1: When #Species is #O, we penalize the most. (Author's own work)

The form 4.1 can be expressed thus in the following form

$$f_{Oimpact} = (1 - 2|\frac{O}{S+O} - 0.5|)(S+O) \tag{4.3}$$

It can be shown by a simple derivation that this formula can be expressed as

$$f_{Oimpact}(S,O) = \begin{cases} 2S & \text{O / (S+O)} \leq 0.5 \\ 2O & \text{otherwise} \end{cases} \tag{4.4}$$

Let us use this formula to assign a value to each inconsistent word in the LINNAEUS. From the Table 4.7 we can see that there are only 20 inconsistent entities in the LINNAEUS. Let us do the same for the S800-orig (table 4.8). In this case, the listing is longer. There are 77 inconsistent entities. Terms such as 'yeast' and *Arabidopsis* have more than 10 tags in both categories. Note that this technique quite well filters out some obvious mistakes. There is one annotation of A in the corpus, hence all the sentences that begin with A are considered missing entities, and their number is abundant. However, since #O is much larger than #Species, this will not dominate in the listing albeit bring that entity to 11th place.

Let us assign a score for a data set with this method. It can be done by summing

all the values and dividing the by the number of species there are in total. Hence,

$$\text{O-impact of a dataset} = \frac{\sum_{\text{S in Species}} f_{Oimpact}(S)}{\#Species} \tag{4.5}$$

Applying the formula on the LINNAEUS and the S800

By giving a score for each data set, we find that (table 4.6):

data set	inconsistency score
LINNAEUS	$\tilde{0}.025$
S800 original	$\tilde{0}.135$

Table 4.6: Inconsistency scores

		$\text{LIN}_{Species}$	LIN_{Os}	LIN_{Oratio}	$\text{LIN}_{f(S,O)}$
1	rodent	3	3	0.500	6.000
2	children	53	2	0.036	4.000
3	Murine	2	2	0.500	4.000
4	cyanobacterial	2	3	0.600	4.000
5	persons	4	1	0.200	2.000
6	moth	1	1	0.500	2.000
7	Scylla	1	1	0.500	2.000
8	shrimp	1	1	0.500	2.000
9	Echinacea	3	1	0.250	2.000
10	Drosophila	20	1	0.048	2.000
11	rat	26	1	0.037	2.000
12	bovine	20	1	0.048	2.000
13	adenovirus	1	7	0.875	2.000
14	murine	63	1	0.016	2.000
15	bacteriophage	1	1	0.500	2.000
16	hamster	1	1	0.500	2.000
17	worms	14	1	0.067	2.000
18	calf	9	1	0.100	2.000
19	Human	13	1	0.071	2.000
20	women	30	1	0.032	2.000
21	Caenorhabditis elegans	4	0	0.000	0.000

Table 4.7: Entities which have both Species and O tags in the LINNAEUS.

Search for hard words

As we know there are some words, whose meaning is ambiguous. For example, 'corn', which we will discuss shortly, can refer to species and genus, and its meaning

		$S800_{Species}$	$S800_{Os}$	$S800_{Oratio}$	$S800_{f(S,O)}$
1	yeast	31	14	0.311	28.000
2	Arabidopsis	14	15	0.517	28.000
3	mice	22	10	0.312	20.000
4	rice	25	10	0.286	20.000
5	human	51	6	0.105	12.000
6	influenza virus	5	7	0.583	10.000
7	Drosophila	8	4	0.333	8.000
8	trout	6	4	0.400	8.000
9	wheat	6	4	0.400	8.000
10	HIV	20	4	0.167	8.000
11	A	4	144	0.973	8.000
11	O	3	14	0.824	6.000
...	...	...	...	...	...
75	tomato plants	1	5	0.833	2.000
76	U. maydis	5	1	0.167	2.000
77	anti-HCV	1	4	0.800	2.000
78	Rumex palustris	1	0	0.000	0.000

Table 4.8: Entities which have both Species and O tags in the S800 original

has to be derived from the context. The method we have used does not consider the context, focusing solely on individual words. This raises an intriguing question: Are there words in the corpus that are prone to ambiguity in their meaning? Luckily, there is an easy way of finding that out. We can look for words that have both the Species-tags and the O-tags in the LINNAEUS and the S800 data sets. Surprisingly, there are only 2 (table 4.9) of them in the whole data set: *adenovirus* and *Drosophila*. So it seems that our method is not perhaps very vulnerable to such systematic errors.

	adenovirus	Drosophila	Streptococcus pneumoniae	Sordaria macrospora	rice
$S800_S$	2	8	1	2	25
$S800_O$	4	4	0	0	10
LIN_S	1	20	1	1	4
LIN_O	7	1	0	0	0
$S800_{Oratio}$	0.666...	0.333...	0.0	0.0	0.2857
LIN_{Oratio}	0.875	0.04761	0.0	0.0	0.0
$LINS800_{Oratios}$	0.1666...	0.06349	0.0	0.0	0.0

Table 4.9: Species, which have mentions Species and Os in both data sets: the LINNAEUS andthe $S800_{orig}$

4.4 Revisioning process

4.4.1 Background: Taxonomical Rank

Tree of life

Let us have a brief recap of the taxonomical rank system. Taxonomical naming of living things was invented by a Swedish botanist, zoologist, taxonomist and physician Carl Von Linné in the 18th century. The Figure 4.2 shows the main divisions: bacteria, archaea, and eukarya. Bacteria and archaea do not have nuclei in their cells. Eukarya is best known as it comprises all mammals. The tree is a human way of classifying living things and it is constantly evolving as new studies are made. The analogy to a tree is that all things that grow from the same branch have a common ancestor and share some common traits from that ancestor.

Traditionally, morphology — which encompasses the study of the form, size, and structure of living organisms — was utilized to classify species within the tree of life. However, in contemporary times, DNA analysis has begun to surpass these traditional methods. A thing to notice is that viruses are not considered living things in the tree of life. However, there exists a classification system also for viruses (see Figure 4.5).

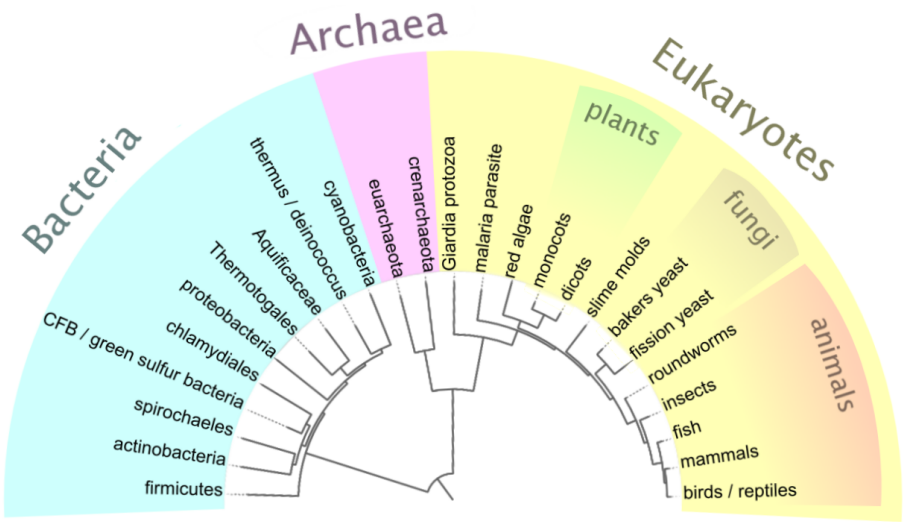

Figure 4.2: Simplified tree of life. Source: Wikipedia, Author: *Madprime*, Licence: CC0

Binomial naming system and NCBI

Binomial nomenclature is the system that was invented by Linné. The first part of the name refers to the genus and the latter to the species. The Figure 4.3 shows the taxonomical rank of the Red fox (*Vulpes vulpes*).

The National Center for Biotechnology Information, NCBI offers a database, which has species names in it. According to their website: *The Taxonomy Database is a curated classification and nomenclature for all of the organisms in the public sequence databases. This currently represents about 10 % of the described species of life on the planet.* NCBI database assigns an ID to every instant in the database and offers a convenient tool for computer scientists. For example, the taxonomical ID for *Vulpes vulpes* is 9627.

Apart from the scientific names, which are universal, there are usually also common names. For example Red fox for *Vulpes vulpes*. Common names however are not always very distinguishing. For example, corn refers can refer to two things: genus alone and also a specific species *Zae mays L.*. So the process of finding all the species in a document cannot be fully automated by a simple search patterns

approach but also the context needs to be considered. This also means that making annotations demands good domain knowledge. Here is the taxonomical Hierarchy of *Zae mays L.* from the NCBI database.

```
Taxonomic Hierarchy


Kingdom: Plantae  > plantes, Planta, Vegetal, plants
Subkingdom: Viridiplantae  > green plants
Infrakingdom: Streptophyta  > land plants
Superdivision: Embryophyta
Division: Tracheophyta  > vascular plants, tracheophytes
Subdivision: Spermatophytina  > spermatophytes, seed plants, phanérogames
Class: Magnoliopsida
Superorder: Lilianae  > monocots, monocotyledons, monocotylédones
Order: Poales
Family: Poaceae  > grasses, graminées
Genus: Zea L. > corn
Species: Zea mays L. > corn
```

Another term that is used in this study is a clade. A clade is a group of species which have a common ancestor. For example siamangs, gibbons, gorillas, chimpanzees and humans all have a common ancestor (see Figure 4.4). In the next section, we will discuss clades in more depth.

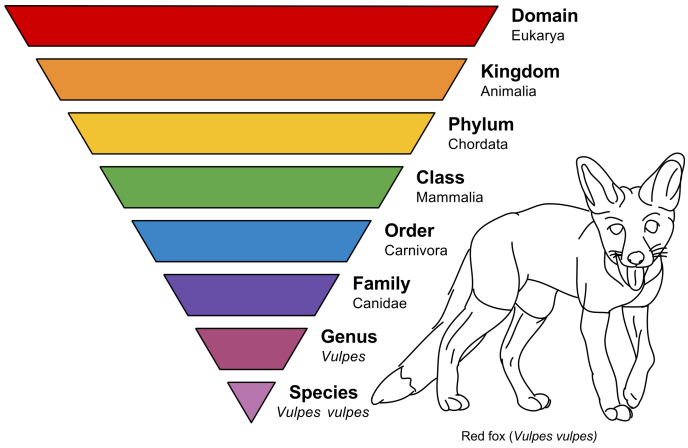

Figure 4.3: This graph shows the main taxonomic ranks: domain, kingdom, phylum, class, order, family, genus, and species. This graph demonstrates how taxonomic ranking is used to designate related animals, the example used here is the red fox (Vulpes vulpes). Source: Wikipedia, Author: *Annina Breen*, Licence: CC BY-SA 4.0

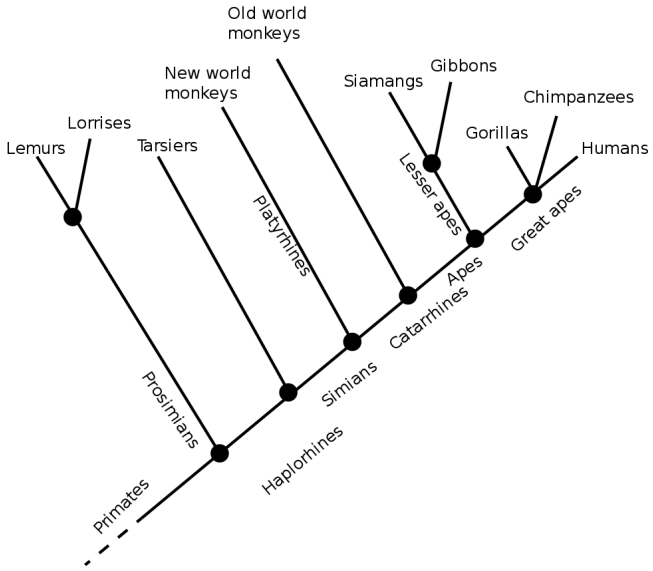

Figure 4.4: Primate Cladogram. Source: Wikipedia, Author:*KDS4444*, Licence:CC0

Viruses

Viruses are not living things and cannot be placed in the tree depicted in 4.2. However, ICTV, International Committee on Taxonomy of Viruses, maintains and develops a taxonomical rank system for viruses. Even though viruses have a hierar-

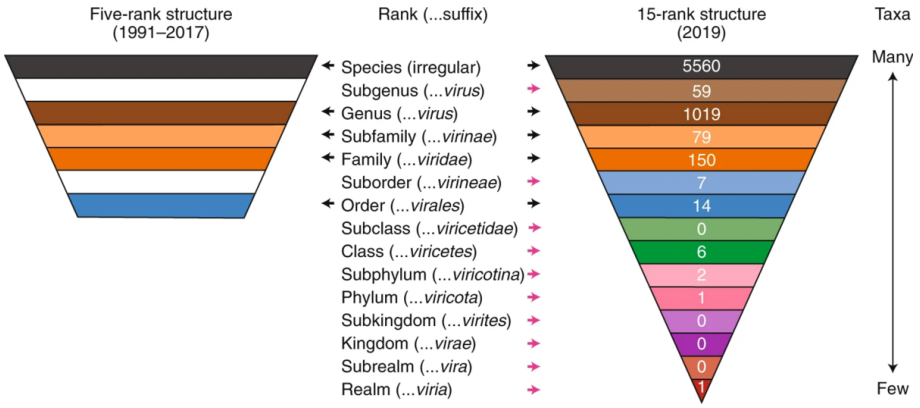

Figure 4.5: Virus taxonomy has hierarchical taxonomy. Source: Wikipedia, Author:*attuale*, Licence: CC BY 4.0

chical rank system there are no standard binomial naming conventions for viruses. As a matter of fact, the naming conventions have changed quite recently, in 2017. (see Figure 4.5) The new system is identical to the one used in flora and fauna. Extending from 5 ranks (species, genus, subfamily, family, and order) to 15 ranks.

Viruses also differ from animals in the sense, that they evolve very rapidly. Every laboratory cultivation of viruses is a new strain and that shows in the nomenclature. Hence it is quite reasonable to think that an automated system for finding species may have problems in particular with viruses.

4.4.2 Guidelines of revisioning

In this section, we try to summarize the revision process. As the process involve making decisions on a large number of small details, we will not try to go through it thoroughly instead it is extracted from Sampo Pyysalo's website on github[21] on 2021-12-11 in appendix A.1.1.

Main principles

1. Decouple recognition from normalization, and address overall consistency

The original idea of the S800 is to tag annotations that can be normalized to NCBI taxonomical ID (of a particular year). However, to make the corpus more suitable for recognizing species and hence account more generally for all species, not only by the strict guidance of NCBI tax-ID, tagging is extended to all species regardless of whether they could be normalized to NCBI ID. If annotations did not fall clearly into species category but rather genera, families and other levels of taxonomy above species, they were marked Out-of-scope.

For example, there are many mentions of 'tree', 'trees', 'hens', 'bears', 'seabird', 'fish', 'trout', 'foxes', 'bug', 'bugs', which seem to be species in common usage of language, but their reference to a specific tx-id is not clear. In this case, the words are marked Out-of-scope.

For building a more general NER system, we might consider also adding some of these to the data set. However, to make the study compared with the original version, we filter these out and be left only with species mentions.

E.g. 20840607.txt Seabird as OOP in this context *2. We compared and projected the population responses of three seabird species living in sub-tropical, sub-Antarctic and Antarctic biomes to predicted*

2. Revise annotated spans for boundary consistency

The original evaluation using the S800 corpus data applied relaxed boundary matching criteria:

[. . .] we used flexible boundary matching of species names, meaning that taggers would receive a true positive if it produced a tag that overlapped with an annotated substring and had the correct assigned taxonomic identifier.

It is quite common that in the original version there are a lot of inconsistencies in boundaries. This is perhaps due to the annotation process, where a large number of people annotate different parts of the corpus independently. Some examples in the Tables 4.10 , 4.11 show that there is no clear consensus on how should 'sp. nov.' prefix be treated.

Serinicoccus	B-Species	Flavobacterium	B-Species
profundi	I-Species	ponti	I-Species
sp	I-Species	sp	I-Species
.	I-Species	.	I-Species
nov	I-Species	nov	I-Species
.	I-Species	.	I-Species

Table 4.10: In some cases sp. nov. is included in tagging in the S800-original

Methanoregula	B-Species	Scardovia	B-Species
formicica	I-Species	wiggsiae	I-Species
sp	O	sp	O
.	O	.	O
nov	O	nov	O
.	O	.	O

Table 4.11: In some cases sp. nov. is excluded in tagging in the S800-original

Guidelines were made to address this problem. The new rule is to remove trailing abbreviations. Some corrected examples are presented in the Table 4.12.

Original	**Corrected**	**Explanations**
SMSP(T)	SMSP	(T) for strain: mark as strain
Flavobacterium ponti sp. nov.	Flavobacterium ponti	species nova - meaning published for first time
Methanoregula boonei gen. nov., sp. nov.	Methanoregula boonei	genus et species nova - meaning published for first time
Caloranaerobacter azorensis	Caloranaerobacter azorensis MV1087	MV1087 strain prefix - meaning published for first time

Table 4.12: Some examples where boundary rules are applied to exclude prefixes

3. Separate strain mentions from species mentions

The original paper that introduced the S800 states that *...lower taxonomic levels (e.g. strain names) were mapped to the NCBI taxonomic identifier of the corresponding species.* The revision process introduced a separate *strain* identifier.

For example in document *19667393* term 'SMSP' was changed from species to strain. and *Methanoregula boonei 6A8* was separated such that *Methanoregula boonei* remained as a species and *6A8* were separated as a strain.

Notice that in building the NER-model other annotated entities than species were filtered out. So this rule will only affect the tagger in the sense that strains are not marked as species.

4. Introduce annotation for upper taxonomic ranks

The original S800 also tagged mentions of names that were higher in the taxonomical rank than species. In some cases, genus names were used to refer to species.

In the revisioned version, *genus* rank was included. For example in the document 19667393 *Methanoregula* was marked as genus: *...properties, we propose that strain SMSP(T) represents a novel species of the genus Methanoregula, for which we propose the name Methanoregula formicica sp. nov., with the type strain.*

Other miscellaneous guidelines

The full list of errors in the original S800 and changes in the annotation is quite exhaustive and cannot be easily summarized. Here is a small subset of examples of those annotations. The full annotation is listed in the appendix A.1.1.

- The corpus features many instances of 'sp. nov.', 'gen. nov.' and 'sp. nov.' after species to indicate, that a new species is introduced in the text. These were left unannotated. Also, forma specialis: 'f. sp.' should not be annotated.
- Person's name, like *Borgmeier* in *Pesudacteon tricuspis Borgmeier* should not be annotated.
- Strain names have sometimes *(T)* appended. Also, this extension is left unannotated.
- Common head nouns, like 'plants', should not be annotated. Nor should premodifiers like 'native'. The same goes for nouns that mark levels in taxonomy, such as 'strain', 'serotype', 'serovar' and 'serogroup'.
- Antibodies premodifier 'anti-', like *anti-HCV* should be annotated.

- In the data set there are **Clades**, which stands for a group of organisms, which are believed to have a common ancestor, in the corpus. These are included in the corpus but are marked in such a way, that they will be assigned to the nearest non-Clade ancestor. This will be evident in *GII.4*, which shows up numerous times in our experiment. The *GII.4* genogroup has around 5000 different strains related to it, and that is only one subtype of one virus. So basically every different laboratory cultivation is a new strain. This is obviously very different compared to the usage of strains in big mammals, where strains are much more uncommon. For example, dogs, which have been subject to breeding show only around 200 different breeds or strains. This, of course, can pose a challenge to the model, as it is not very obvious how should so different concepts such as dogs and viruses fit under the same taxonomic naming.

- rat: synonym for *Rattus norvegicus* which is also known as the brown rat, the common rat, street rat, sewer rat, wharf rat, Hanover rat, Norway rat, Norwegian rat and Parisian rat and Rattus (genus). Should be annotated as *Rattus norvegicus* with taxid:10116, unless explicitly referring to a different taxonomic unit (for example cotton rat: taxid:42414 (genus))

- as a rule of thumb, if it is not clear in the context whether a word refers to a species or a higher-level level entity, it should be annotated as species by default.

- Common names like human, horse and rats should always be annotated.

- yeast: If a span includes yeast and refers to a higher taxonomical rank than species, it should be marked as Out-of-scope.

4.5 A look after the revisions: S800-orig vs. S800-rev

To run the same analysis on the revised version of S800, we get the results in Table 4.13. We can see that the total number of entities having both Species and Os are now much lower, only 10. Secondly, the top entity has only 3 Os. It is also interesting to see, that there are no common entities in the TOP-10 listings of original and revision versions (see Table 4.8).

		$S800_{Species}$	$S800_{Os}$	$S800_{Oratio}$	$S800_{f(S,O)}$
1	NDV	14	3	0.176	6.000
2	SIV	2	2	0.500	4.000
3	morels	2	2	0.500	4.000
4	cotton	3	2	0.400	4.000
5	tomato	22	2	0.083	4.000
6	swine	2	1	0.333	2.000
7	chicken	4	1	0.200	2.000
8	bee	1	3	0.750	2.000
9	Lassa virus	5	1	0.167	2.000
10	M . australis	5	0	0.000	0.000

Table 4.13: Entities which have both Species and O tags in the S800 revision

After applying the previous method, which analyzed controversial entities, to the revised version of S800, we discovered that it significantly outperforms the original version. The comparative scores can be found in Table 4.14.

data set	inconsistency score
LINNAEUS	$\tilde{0}.025$
S800 original	$\tilde{0}.135$
S800 revised	$\tilde{0}.013$

Table 4.14: Inconsistency scores

4.5.1 Word level studies on the original and the revisioned

Let us compare the difference between the two datasets: original and revisioned. [1] The Figure 4.6 describes the word distribution by their frequency. The X-axis describes the number of times an entity is expressed in the corpus and the Y-axis percentage how much these observations contribute to the total number of annotations. For example in the case of the original data set (blue line), there are 684 species which occur only one time, such as *Ugandan cassava brown streak virus*, these makeup around 22.5 % of all the species annotated in the corpus and by comparison there is one entity, specifically 'human', which is present 94 times, which consist of around 3 % of all the annotated mentions. Some observations: The original data set has more tagged words that occur only once in the document. That trend holds till the end. The number of the most frequent words has gone up, suggesting these words have been left out accidentally from the first version resulting in inconsistencies. Most notably 'human', which has 12 more marks (94 to 106) and 'mice', which has 21 more tags (31 to 52). Another thing worthwhile noticing is that entities which occur 1,2 or 3 times consist of around 50 % of all the mentions. Entities which occur up to around 15 times consist of roughly 80 % of all mentions.

[1] To make the two datasets more comparable the following rules were applied: the space was omitted on one of the sides on the following: " . " to ". ", " (" to " (", ") " to ") ", " - " to "-" and " ' " to "' "

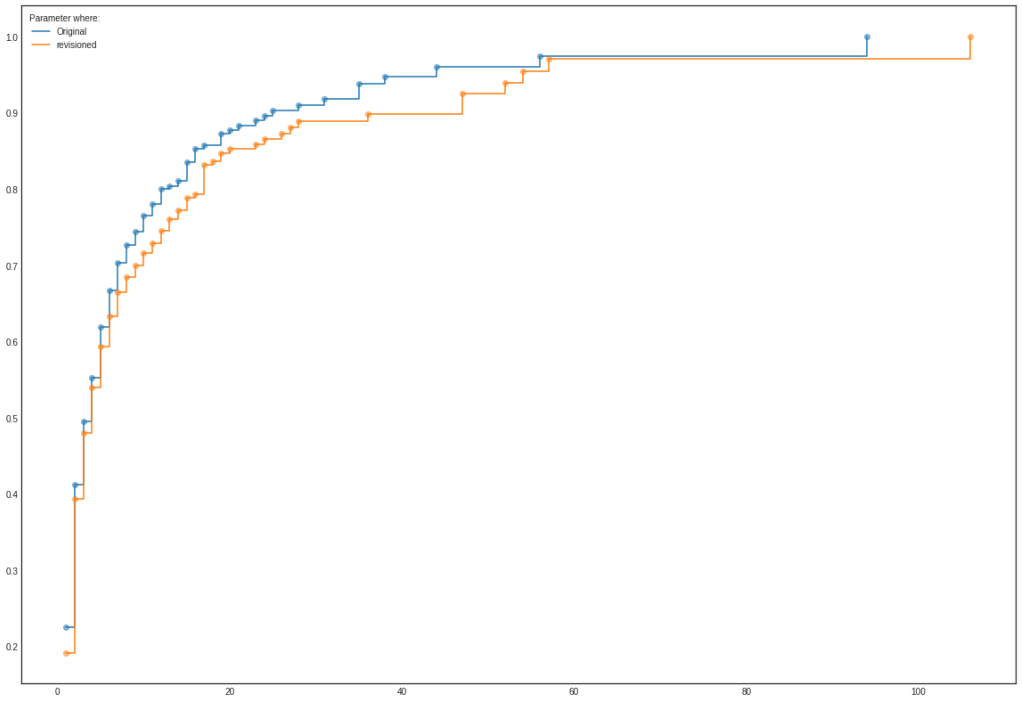

Figure 4.6: Word occurrence cumulative distribution. See the full listing in the appendix A.1

A look at the most frequent words

The TOP-10 words in Table 4.15 make up 480 tags (revisioned) and 406 tags (original) in the corpora and make up around 11 % of the all the tags, therefore this group is not insignificant and the decisions to rule out some of the most frequently used species mentioned such as 'patient', 'yeast', 'man', may result in significant performance issues of the model.

Table 4.15 shows some of the top words in the two data sets. Here we can see a trend of inclusion of the most frequent words. Only one of the top words has seen a decline in tagging: 'yeast', which was ruled to be non-tagged. There are many instances where the number of tags has grown more than 10 %. Most notably 'mice', which is up to 67 %. Also word 'human', which makes up around 2.5 % of the tags altogether has seen a sharp increase.

	word	**In original**	**In revisioned**	**Change**	**Change in %**
1	human	94	106	12	12.77
2	HIV-1	56	57	1	1.79
3	HIV	38	54	16	42.11
4	Escherichia coli	44	47	3	6.82
5	mice	31	52	21	67.74
6	rice	35	47	12	34.29
7	HCV	28	36	8	28.57
8	maize	25	28	3	12.00
9	mouse	23	27	4	17.39
10	Aspergillus nidulans	24	24	0	0.00
11	wheat	19	26	7	36.84
12	tomato	21	23	2	9.52
13	Saccharomyces cerevisiae	20	20	0	0.00
14	Aspergillus fumigatus	19	19	0	0.00
15	MV	19	19	0	0.00
16	yeast	35	0	-35	-100.00
17	Candida albicans	17	17	0	0.00
18	HDV	16	17	1	6.25
19	C. albicans	16	17	1	6.25
20	Bacillus subtilis	16	17	1	6.25
21	HBV	15	17	2	13.33
22	tobacco	15	17	2	13.33
23	VZV	15	17	2	13.33
24	A. nidulans	16	16	0	0.00

Table 4.15: Some of the most frequent words in the two versions of corpora.

Analyzing the rarest words

If we take into account only the rarest species tags, those which occur only once in the document and make up around 20 % of all the tags, and divide them by words — that is, splitting them by whitespaces — we get the distribution shown in the Figure 4.7. In the Table 4.16 there is one example from all of the categories.

It seems that the rarest species tags in revisioned data have two words: e.g:*wax moth*, *Vrihiamoeba italica* and *V. paradoxus*. They make up more than 75 % of this category. In the case, the original dataset, their distribution is more spread out. This is a sign of better quality as the binomial nomenclature uses a two-word latin name consisting of the genus name and the species name.

	Original	**Revisioned**
1	YHV	YHV
2	yellow morels	Xylella fastidiosa
3	Zea mays (L.)	yellow head virus
4	Xanthomonas oryzae pv. oryzae	white spot syndrome virus
5	Ugandan cassava brown streak virus	Ugandan cassava brown streak virus
6	swine-origin influenza A virus (S-OIV) (H1N1)	Marek' s disease virus type 1
7	F. oxysporum f. sp. lycopersici isolate FGSC 9935	type A H1N1 and type B influenza viruses

Table 4.16: Examples of splitting the rares tags by whitespace

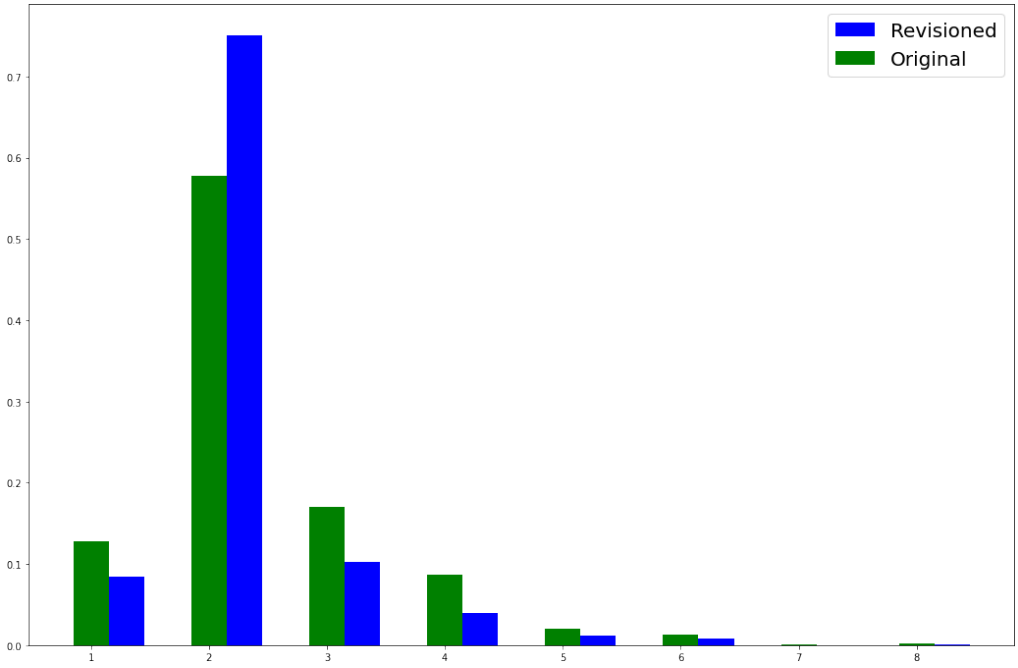

Figure 4.7: Species, which has only one tagged mention, by their size (number of whitespaces)

A qualitative look at the rarest words

As was pointed out earlier the proportion of species tags, — which has only one mention in the corpora — has raised around 3.4 % points (19.15 vs 22.55). We take a look at some of the words that have been added and left out in the revision process. They are presented in the appendix A.1.1. The focus is on tags that are in the most prominent category, composed of two words. One of the most evident differences is the exclusion of strains appended by *(T)*. Moreover, a lot of the left-out tags look cumbersome, having abbreviations, digits, hyphens, etc., whereas the tags that were added all seem to have a scientific form.

4.5.2 A few words about the LINNAEUS

The problem of leaving out

One thing that is noticeable in the LINNAEUS is that it contains common names such as patients, which are mostly filtered out in the S800 dataset. There is ongoing debate about whether those entities should be added or left out, with arguments both for and against. On one hand, these terms represent species: for example, 'patient' refers to *Homo sapiens*, which possesses an NCBI taxonomy ID. On the other hand, it is crucial to differentiate between instances where common names directly refer to the species itself and instances where they primarily serve as subjects in sentences, such as in references to 'a boy' or 'a fisherman'. However, we won't go into the details of this debate.

The TOP-20 words are listed in Table A.2. Some of the words play major role in the LINNAEUS. Some of these words are listed in A.2. 'patients' have 437 occurrences, which makeover 10 % of all the species tags in the corpus. When the singular form 'patient' is also added to that, they together make almost 15 % of the tags in the corpus. These are big numbers and we can see in Figure the 4.8 and the Table A.2 that the 15 most common species names make up around 50 % of the overall species tags.

All in all, for better comparison purposes, we make an alternative version of the LINNAEUS, where we filter out most of the common names. The following species were left out: 'boys', 'children', 'Children', 'girls', 'infants', 'infants', 'men', 'Men', 'participant', 'Participant', 'participants', 'Participants', 'patient', 'Patient', 'patients', 'Patients', 'people', 'People', 'peoples', 'person', 'persons', 'Persons', 'schoolchildren', 'woman', 'women' and 'Women'.

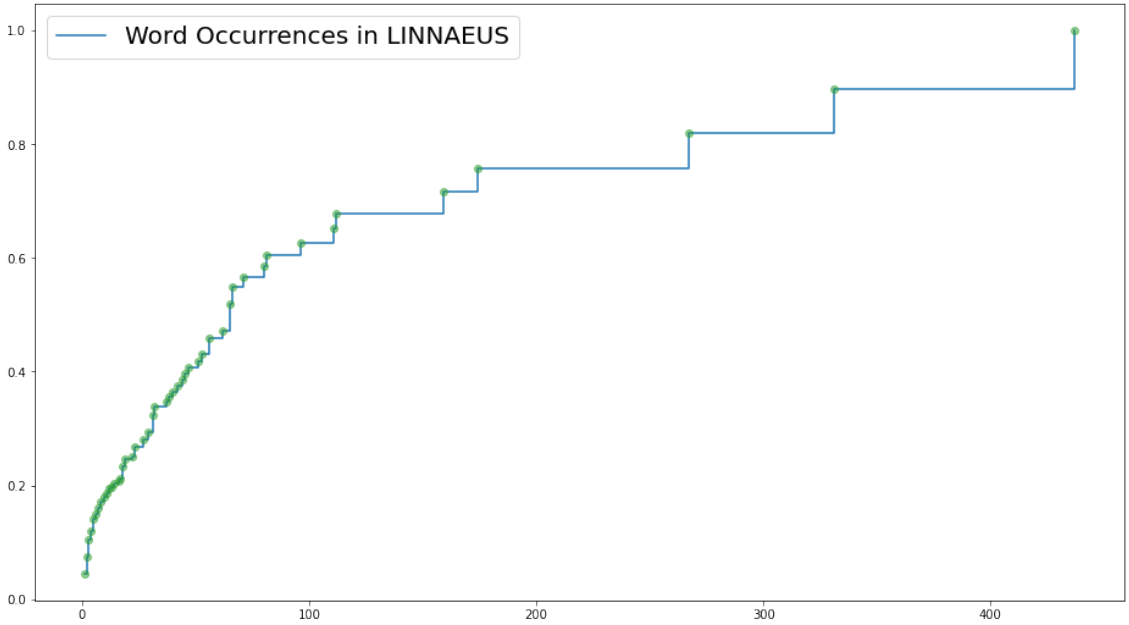

Figure 4.8: The LINNAEUS data set word frequency distribution

5 Results

5.1 Parameter settings for the runs

The performance of the BERT model can vary significantly based on the selection of parameters. Given the infinite number of possible parameters and the impracticality of evaluating them all, we adopted the recommended settings outlined in the BERT paper [10]. Each experiment was conducted using the following hyperparameters:

Maximum Sequence Length	128 256 320 400
batch size	4 8 16
learning rate	5e-5 5e-6
Iterations	3 times each run and then average
Trainfile	train.tsv
Testfile	devel.tsv
Evaluationfile	test.tsv
Maximum number of epochs	6 (early stopping is used)
Optimizer	Adam
Model	biobert v1.1 pubmed

As there are many different models (different sets of parameter combinations), we used the following approach to find the best model and its performance:

1. A grid is formed from all available parameters. Making up $4 * 3 * 2 = 24$

different combinations

2. Using each parameter combination model training is done using train.tsv for training loss and devel.tsv for validation loss. These runs are performed 3 times and the average of the F1-scores is calculated. Early stopping is used in this phase. If both train and test losses continue to decrease the training process may go maximum of 6 epochs.

3. These so called 'devel'-runs are ranked by average (from 3 runs).

4. From these runs, the best model is selected and using it, the final F1-score is calculated using test.tsv data set.

As we are studying both the LINNAEUS and the S800 and since both of them consists of 3 different dataset — train, dev and test — we can make many different combinations from these dataset. We use the following term:

1. **In-corpus** Train and devel sets come from same dataset — that is e.g LINNAEUS itself.

2. **Cross-corpus** Training data set is selected from another model and devel from the other. E.g. train.tsv from the LINNAEUS and devel.tsv from the S800

3. **Combi-cross corpus** Training data set is selected by combining training data sets from both data sets and devel from the other. E.g train.tsvs from the LINNAUES and the S800 and devel.tsv from the S800.

In the following sections we will take a look at devel-runs. In in-corpus runs we present histogram of F1 results and in each run we list the most common false negatives and false positives that get some insight of the taggings. In the latter part we take a look at the evaluation performance of these data sets and analyse their performance also in terms of the S800 document category.

5.2 In-corpus runs

In this section we will examine in-corpus runs. That is runs in which training set, devel set and test set all come from the same data set.

5.2.1 LINNAEUS original and filtered

The LINNAEUS devel set offers extremely good results. Figure 5.1 shows that most of the F-scores from the grid falls between 96 % and 97 %. Figure in appendix A.2 shows similar pattern for the LINNAEUS filtered, but the overall scores tend to be around 1 pp. less, best model having F-score of 96.54 % (table 5.3). Table 5.1 shows the top results from the run. The best parameter combination is with maximum sequence length 512 as with all of the runs, batch size 2, learning rate 3e-5 and only 2 epochs. The Table shows that 3 iterations were conducted on each run and their average was used to determine the performance of the parameter combination. Some of the parameter combinations seem to be stable than other. That can be examined by STDEV column at the rightmost part of the Table. In this case the last run seems to be most stable with STDEV 0.08. Table 5.2 shows that somehow the term 'S. FAECIUM' is missed by the model and interestingly if the data set is filtered the term is captured (see Table 5.4), even tough it does not seem to be a common name. So in the case of the LINNAEUS original at least one seventh of FNs comes from a single document and a single term.

Interestingly looking at Tables 5.2 and 5.4 we can see it is mostly common names that occur in both list: FN and FP. Reasonable explanation for this is that the tagging of the entities is not very consistent. As we can see there is women in the LINNAEUS FP listing, which is clearly a mistake. An interesting extended study of this, would be to find out how well models do in terms of general names versus scientific names.

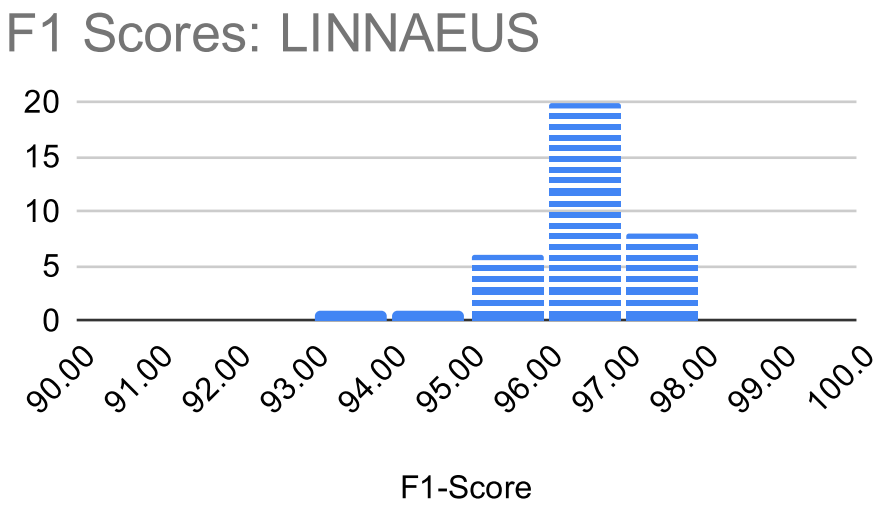

Figure 5.1: The LINNAEUS corpus - devel.tsv: Histogram of the F1-scores.

Grid setting		MSL=512	**Performance**				**F-score**	
BS	**LR**	**Epochs**	**Acc.**	**Precision**	**Recall**	**F1**	**AVG**	**STDEV**
2	3e-5	2/6	99.95	97.72	97.03	97.38	97.12	0.28
2	3e-5	2/6	99.95	97.99	96.33	97.15	97.12	0.28
2	3e-5	2/6	99.94	96.62	97.03	96.83	97.12	0.28
4	5e-5	2/6	99.95	97.58	96.89	97.24	97.04	0.34
4	5e-5	2/6	99.95	97.72	96.75	97.23	97.04	0.34
4	5e-5	2/6	99.94	95.71	97.60	96.64	97.04	0.34
4	2e-5	2/6	99.95	96.51	97.60	97.05	96.97	0.08
4	2e-5	2/6	99.95	96.90	97.03	96.97	96.97	0.08
4	2e-5	2/6	99.95	96.76	97.03	96.90	96.97	0.08

Table 5.1: The LINNAEUS corpus - devel.tsv: TOP3 runs from the grid run.

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
3 S . FAECIUM	3 S . FAECIUM	2 grapevines	2 grapevines
2 person	2 person	2 bacteriophage	2 bacteriophage
2 bananas	2 bananas	1 women	1 women
1 yeast	1 Streptococcus Faecium	1 T . aestivum	1 personvern
1 T . aestivum var Fortal	1 salmon	1 personvern	1 Oenothera
21 total	**18 total**	**16 total**	**13 total**

Table 5.2: The LINNAEUS corpus - devel.tsv: Some samples with their frequencies of false negatives and false positives

Grid setting		MSL=512	**Performance**				**F-score**	
BS	**LR**	**Epochs**	**Acc.**	**Precision**	**Recall**	**F1**	**AVG**	**STDEV**
2	2e-5	2/6	99.95	97.54	95.56	96.54	96.34	0.34
2	2e-5	2/6	99.95	97.90	95.19	96.53	96.34	0.34
2	2e-5	2/6	99.95	95.59	96.30	95.94	96.34	0.34
8	5e-5	3/6	99.96	95.97	97.04	96.50	96.27	0.23
8	5e-5	2/6	99.95	97.17	95.37	96.26	96.27	0.23
8	5e-5	2/6	99.95	95.60	96.48	96.04	96.27	0.23
4	5e-5	2/6	99.95	97.18	95.56	96.36	95.96	0.42
4	5e-5	2/6	99.95	96.80	95.19	95.99	95.96	0.42
4	5e-5	2/6	99.94	96.24	94.81	95.52	95.96	0.42

Table 5.3: The LINNAEUS corpus filtered/ - devel.tsv: TOP3 runs from the grid run.

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
3 Potato	3 Potato	2 grapevines	2 grapevines
2 mules	2 mules	2 bacteriophage	2 bacteriophage
2 man	2 man	1 T . aestivum	1 S
2 bananas	2 bananas	1 S	1 Eucalyptus
1 yeast	1 S . FAECIUM	1 fission yeast	1 Drosophila
24 total	**21 total**	**13 total**	**10 total**

Table 5.4: The LINNAEUS corpus filtered/ - devel.tsv: Some samples with their frequencies of false negatives and false positives

5.3 S800

There is clear difference between original and revisioned data sets. Histogram of the S800 devel runs in Figure 5.2 shows nice bell curve-like shape, where the results span across 69 % to 76 %, with majority of the runs settling at 73 %. Figure 5.3 on the other shows that the S800-rev devel set gives extremely good F-score results, worst being around at 92 % and the best at 97 %. The best results is achieved with batch size 2, learning rate 3e-5 and only 2 epochs (table 5.6). As with the S800-orig and the LINNAEUS also the S800-rev performs best at batch size 2 and 2 epochs. Only this time learning rate is 5e-5. It seems that batch size of 2 offers the best models. If this holds true in subsequent runs, the grid could be adjusted to only use batch size 2. From FNs in Table 5.7 we can see that in the case of the S800-orig terms such as 'SIVmac239', 'HIV' and 'VSV' are missed many times, hinting that at least some of them come from a single document. The FPs are hard to interpret as they seem to be correct, as 'VSV - G' and 'KM 3842 (T)'. It seems that they might have been missed by the annotator. Table 5.8 shows on the other hand that FNs and FPs have become much more clear. Clumsy abbreviations are not present. It is interesting to see that common names ('cacao', 'bees') are most frequent in FN listing. The fact that there are only 1 FP in any overlap, is quite staggering.

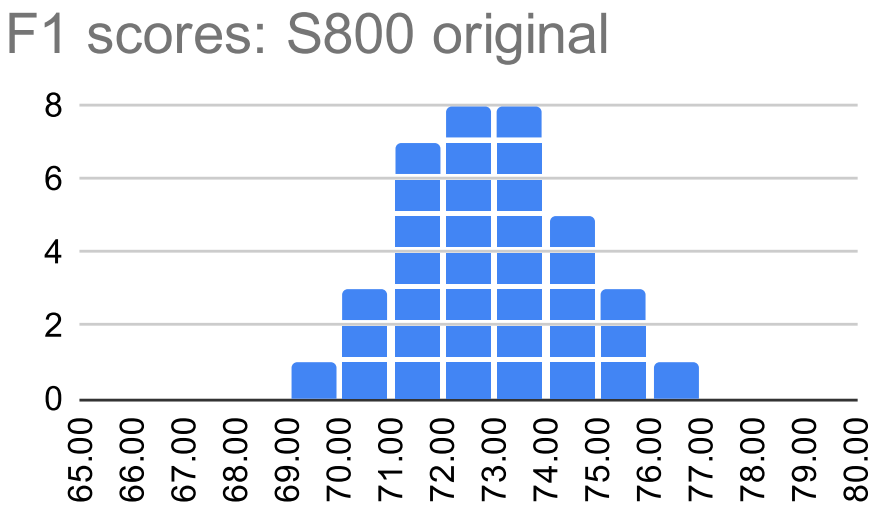

Figure 5.2: S800 original: Histogram of the F1-scores.

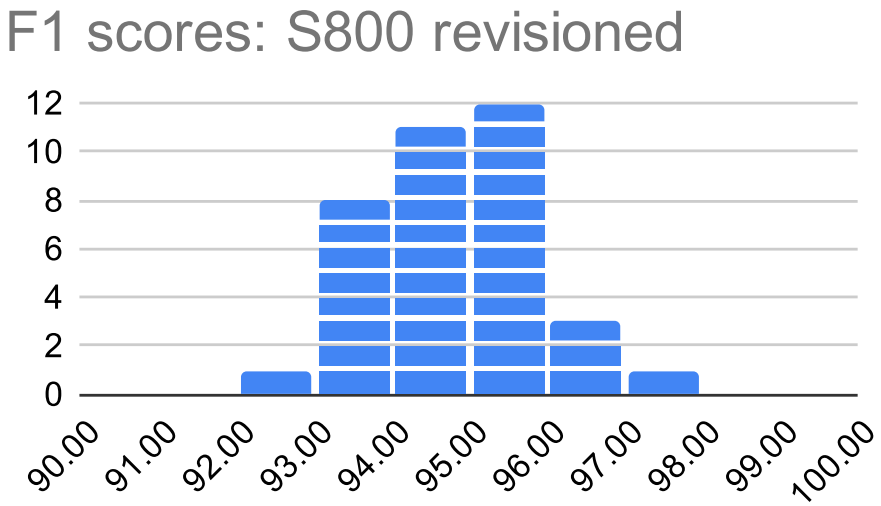

Figure 5.3: S800 revisioned20-only-species - devel.tsv: Histogram of the F1-scores.

Grid setting		MSL=512	**Performance**				**F-score**	
BS	**LR**	**Epochs**	**Acc.**	**Precision**	**Recall**	**F1**	**AVG**	**STDEV**
2	5e-5	2/6	98.83	76.10	76.30	76.20	74.49	2.21
2	5e-5	2/6	98.83	76.90	73.70	75.27	74.49	2.21
2	5e-5	2/6	98.60	72.37	71.61	71.99	74.49	2.21
16	3e-5	2/6	98.82	73.28	75.00	74.13	73.92	0.35
16	3e-5	2/6	98.81	72.28	76.04	74.11	73.92	0.35
16	3e-5	2/6	98.89	74.40	72.66	73.52	73.92	0.35
16	2e-5	2/6	98.90	76.13	74.74	75.43	73.55	2.16
16	2e-5	2/6	98.88	74.60	73.44	74.02	73.55	2.16
16	2e-5	2/6	98.77	68.86	73.70	71.19	73.55	2.16

Table 5.5: S800 original-only-species/ : TOP3 runs from the grid run.

Grid setting		MSL=512	**Performance**				**F-score**	
BS	**LR**	**Epochs**	**Acc.**	**Precision**	**Recall**	**F1**	**AVG**	**STDEV**
2	3e-5	2/6	99.89	98.78	95.31	97.01	95.97	0.97
2	3e-5	2/6	99.82	97.57	94.13	95.82	95.97	0.97
2	3e-5	2/6	99.82	96.67	93.55	95.08	95.97	0.97
4	5e-5	2/6	99.86	96.48	96.48	96.48	95.68	0.70
4	5e-5	2/6	99.82	93.77	97.07	95.39	95.68	0.70
4	5e-5	2/6	99.81	95.03	95.31	95.17	95.68	0.70
8	5e-5	2/6	99.86	95.94	97.07	96.50	95.68	0.86
8	5e-5	2/6	99.83	95.61	95.89	95.75	95.68	0.86
8	5e-5	2/6	99.82	96.36	93.26	94.78	95.68	0.86

Table 5.6: S800 revisioned20-only-species - devel.tsv: TOP3 runs from the grid run.

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
7 SIVmac239	7 SIVmac239	4 VSV - G	3 .
6 HIV	4 elephant	3 R5 HIV	2 Valsa melanodiscus
5 VSV	4 Bdellovibrio	3 KMM 3842 (T)	2 T
4 elephant	3 viral	3 HIV R5	2 mice
91 total	**50 total**	**92 total**	**48 total**

Table 5.7: S800 original-only-species/ : Some samples with their frequencies of false negatives and false positives

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
5 cacao	5 cacao	1 . scandens	1 FVB
2 bees	2 bees	1 ouzel	
1 VSV	1 VSV	1 FVB	
1 SIV	1 SIV	1 Australian grey - crowned babbler	
4 total	**16 total**	**13 total**	**1 total**

Table 5.8: S800 revisioned20-only-species - devel.tsv: Some samples with their frequencies of false negatives and false positives

5.4 Cross-corpus and combi-cross experiments

In the following experiments we are combining the training data sets and evaluating them against development dataset. Cross-corpus experiment simply swaps the development set from another data set and combi-cross experiment combines the training sets first and then evaluates the model against one of the devel data sets.

5.4.1 Train: S800 Devel: LINNAEUS

We can see that in both cases: the S800 original and revisioned, the filtered version does much better than the original one — see Table 5.9 and 5.10. The gap between F-scores is around 10 % points. A look at the false negatives section at the Tables suggest that indeed the filtered terms — such as 'patient', 'patients' — are a cause for this performance level up. Furthermore, the revision process has resulted in an improvement of 3 percentage points in the original LINNAEUS dataset and 7 percentage points in the filtered LINNAEUS dataset. An F-score of 92.74 % is considered excellent by any standard. The S800 revisioned misses term 'yeast' 19 times (table 5.10), which is around half of the FNs.

In this run we can see clearly that filtering has done a good job in making the LINNAEUS more compatible with the S800 and indeed this must be a significant difference between the two data sets : leaving out or including common names. Table 5.10 shows that F-score with the LINNAEUS filtered: 92.74 % is quite impressive, where as the original results 80.29 % is not.

Train: S800 original Devel: LINNAEUS

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 77.01 % (78.41 %)	
69 patient	69 patient	9 enterococcal	9 enterococcal
53 patients	53 patients	7 HK	7 HK
13 wheat	11 wheat	6 91	6 91
214 total	**205 total**	**81 total**	**72 total**
LINNAEUS filtered		F-score: 85.37 % (86.37 %)	
5 Human	5 Human	16 MDA	16 MDA
4 wheat	3 wheat	12 HK	12 HK
3 mice	3 mice	8 enterococcal	8 enterococcal
44 total	**35 total**	**126 total**	**117 total**

Table 5.9: Train: S800-orig Devel: LINNAEUS : F1-scores and TOP FNs and FPs

Train: S800 revisioned Devel: LINNAEUS

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 80.29 % (81.10 %)	
69 patient	69 patient	5 brevis	5 brevis
53 patients	53 patients	4 HK	4 HK
19 yeast	19 yeast	3 enterococcal	3 enterococcal
211 total	**206 total**	**33 total**	**28 total**
LINNAEUS filtered		F-score: 92.74 % (94.23 %)	
19 yeast	19 yeast	4 HK	4 HK
2 sheep	2 man	2 grapevines	2 grapevines
2 man	2 Drosophila	2 fetal sheep	2 Enterococcal
42 total	**34 total**	**36 total**	**28 total**

Table 5.10: Train: S800-rev Devel: LINNAEUS : F1-scores and TOP FNs and FPs

5.4.2 Train: LINNAEUS Devel: S800

When the two sets of the LINNAEUS are used in training and the S800 devel in testing, the similar pattern as we saw previously holds. Revisioned version does better and so does the filtered version. Revisioning and filtering brings around 10 % points advantage. Normal version of the LINNAEUS treats 'patient(s)' as positives which is presumable. Contrary to previous study now we can see non common names in FN listing. Table 5.11 shows a lot of gibberish abbreviations in FN listings, they also appear in any overlap sections, indicating that the LINNAEUS-trained-model misses them completely. They appear to come from a same document. That means a single document has a large impact on F-score. This also happens with the revisioned S800: the model tags 'SIV*' terms as positives (table 5.12). It is likely that those words come from one article, as the S800 is full of abstracts.

Train: LINNAEUS Devel: S800 original

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 71.84 % (81.00 %)	
6 KMM 3851	6 KMM 3851	6 patients	6 patients
6 KMM 3842	6 KMM 3842	5 S . singularis	5 S . singularis
6 15 - Je - 017 (T)	6 15 - Je - 017 (T)	4 patient	4 patient
4 MV	4 MV	3 rodent	3 rodent
97 total	**65 total**	**128 total**	**87 total**
LINNAEUS filtered		F-score: 74.71 % (83.61 %)	
6 KMM 3851	6 KMM 3851	5 S . singularis	5 S . singularis
6 KMM 3842	6 KMM 3842	4 Entamoeba	4 Entamoeba
6 15 - Je - 017 (T)	6 15 - Je - 017 (T)	3 Mantamonas	3 Mantamonas
4 VSV	4 cacao	2 wheat	2 wheat
99 total	**67 total**	**94 total**	**58 total**

Table 5.11: Train: LINNAEUS Devel: S800-orig : F1-scores and TOP FNs and FPs

Train: LINNAEUS Devel: S800 revisioned

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 81.77 % (85.76 %)	
7 MV	7 MV	7 SIVmac239	7 SIVmac239
5 VSV	5 cacao	6 patients	6 patients
5 cacao	4 VSV	4 patient	4 patient
3 EBV	3 EBV	3 Arabidopsis	3 Arabidopsis
54 total	**41 total**	**74 total**	**59 total**
LINNAEUS filtered		F-score: 85.09 % (90.60 %)	
5 VSV	5 VSV	7 SIVmac239	7 SIVmac239
5 cacao	5 cacao	4 elephant	4 elephant
2 sycamore lace bug	2 sycamore lace bug	3 SIVmac	3 SIVmac
2 P falciparum	2 MV	3 SIVcpz	3 SIVcpz
47 total	**29 total**	**56 total**	**36 total**

Table 5.12: Train: LINNAEUS Devel: S800-rev : F1-scores and TOP FNs and FPs

5.4.3 Train: LINNAEUS and S800 Devel: S800

In this case we combine the training data of the LINNAEUS and the S800 and test them against the S800 — see Tables 5.13 and 5.14 for reference. The results are in line with the previous section in that revisioning has made an improvement.

The interesting thing here to notice is that filtering does decrease the performance in the S800 revisioned case. The gap is substantial 13 pp (from 95.58 % to 83.36 % Table 5.14). Both FNs and FPs have raised and terms such as *Bdellovibrio, Entamoeba, Arabidopsis* are frequent in FPs. However there seems not to a single document — as was previous case 'SIV*' in the Table 5.12 — which could explain this. In fact this finding is hard to explain and suggests that the normal version of the LINNAEUS is more coherent as we might think, and simple filtering may destroy some of that coherency.

Train: LINNAEUS and S800 original Devel: S800 original

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 76.310 % (87.33 %)	
6 SIVmac239	6 SIVmac239	9 OsEDR1	9 OsEDR1
5 VSV	3 viral	4 VSV - G	3 S . singularis
5 HIV	3 SIVsm	3 S . singularis	3 nov
86 total	**45 total**	**99 total**	**54 total**
LINNAEUS filtered		F-score: 77.68 % (89.41 %)	
6 SIVmac239	6 SIVmac239	3 R5 HIV	2 wheat
5 elephant	5 elephant	2 wheat	2 Valsa melanodiscus
5 Bdellovibrio	5 Bdellovibrio	2 VSV - G	2 R5
90 total	**49 total**	**79 total**	**31 total**

Table 5.13: Train: LINNAEUS and S800-orig Devel: S800-orig : F1-scores and TOP FNs and FPs

Train: LINNAEUS and S800 revisioned DEVEL S800 revisioned

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 95.58 % (97.47 %)	
3 cacao	3 cacao	1 herpes simplex virus type	1 FVB
2 bees	2 bees	1 FVB mice	1 dwarf bamboo
1 VSV	1 VSV	1 FVB	1 Cytospora canker
1 Theobroma Cacao	1 Theobroma Cacao	1 dwarf bamboo	1 Bdellovibrio
16 total	**13 total**	**7 total**	**4 total**
LINNAEUS filtered		F-score: 83.36 % (90.45 %)	
6 MV	6 MV	6 Bdellovibrio	6 Bdellovibrio
4 cacao	4 cacao	4 Entamoeba	4 Entamoeba
3 VSV	3 VSV	3 Arabidopsis	3 Arabidopsis
2 Variovorax paradoxus	2 Bt	2 yeast	2 yeast
53 total	**29 total**	**62 total**	**37 total**

Table 5.14: Train: LINNAEUS and S800-rev Devel: S800-rev : F1-scores and TOP FNs and FPs

5.4.4 Train: LINNAEUS and S800 Devel: LINNAEUS

In case of combining the data sets and testing them against the LINNAEUS, all of the F-score results are more or less the same — see Tables 5.15 and 5.16. The FNs listing of the Table 5.16 shows 'yeast' as mutual term with the LINNAEUS original and filtered. Apart from that, there seem to be no mutual term between the runs.

These results are interesting and quite contradictory to the previous results where there was a huge difference. Revisioning process does not help the model. A more relevant point of view is perhaps that mixing the S800-orig with the LINNAEUS does not really affect the performance of the LINNAEUS. The in-corpus results of the LINNAEUS yielded 97.12 % (Table 5.1) and the S800 against the LINNAEUS 77.01 % — Table 5.9 keeping in mind that terms 'patient' etc. make up a great deal of this effect, so it would be probable that mixing these two training sets would results the average of them, around 87 %. However, that is not the case. The top F-score results worsens only a few percentage points. So the worse model does not affect the best model a lot. Another point of view is the quality of the data set. Could the LINNAEUS devel and train sets be too similar? In other words does devel leak information to train in the LINNAEUS.

Train: LINNAEUS and S800 original Devel: LINNAEUS

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 96.05 % (96.90 %)	
2 Potato	2 person	5 enterococcal	5 enterococcal
2 person	2 mules	2 grapevines	2 grapevines
2 mules	2 calf	2 brevis	2 brevis
30 total	**27 total**	**9 total**	**6 total**
LINNAEUS filtered		F-score: 96.32 % (96.88 %)	
5 wheat	5 wheat	2 grapevines	2 grapevines
3 S . FAECIUM	3 S . FAECIUM	1 simian virus	1 simian virus
2 man	2 man	1 fission yeast	1 Eucalyptus
27 total	**21 total**	**29 total**	**23 total**

Table 5.15: Train: LINNAEUS and S800-orig Devel: LINNAEUS : F1-scores and TOP FNs and FPs

Train: LINNAEUS and S800 revisioned Devel: LINNAEUS

False negatives		**False positives**	
Exact match	Any overlap	Exact match	Any overlap
LINNAEUS original		F-score: 96.52 % (97.37 %)	
5 yeast	4 yeast	2 HK	2 HK
2 person	2 person	2 hamster	2 grapevines
2 patients	2 patients	2 grapevines	2 Enterococcal
28 total	**22 total**	**21 total**	**15 total**
LINNAEUS filtered		F-score: 96.08 % (96.83 %)	
4 yeast	3 yeast	2 Enterococcal	2 Enterococcal
2 man	2 man	1 Streptococcus	1 simian virus
2 calf	2 calf	1 simian virus	1 Rosevalt
25 total	**21 total**	**17 total**	**13 total**

Table 5.16: Train: LINNAEUS and S800-rev Devel: S800-rev : F1-scores and TOP FNs and FPs

5.5 Test runs on the test.tsv

In this section, we pick up the best-performing models that we found in the previous experiment and test them against their relative test sets. In this way, we can assign a final score to each model. Note that in some of these experiments, we have modified both the training and test sets by using the S800 dataset for both. However, in other cases, we have only altered the training set, utilizing the S800 dataset for training and the LINNAEUS dataset for testing.

5.5.1 In-corpus experiment

Train: S800 Test: S800

We can see that F-score has improved a lot, ∼17 % points with the test-set — see Tables 5.17 and 5.18. The F-score of 89.49 % is perhaps less than we hoped for, as the data performed so well with the development data set. From Tables 5.17 and 5.18 we can see that the improvement has gone up especially in precision, going up ∼25 % points. It is worth noticing that the recall score of any overlap criteria hasn't practically improved, the diff is only ∼0.8 % point. So that is to say, that our new model may be better at finding where exactly the entity begins and ends but it hasn't found any new relevant items. On the other hand, our new model is much better now leaving out false positives.

In summary, previously the model was eager to tag species and it accidentally tagged too many of them, but in spite of that was roughly able to get all of the species picked. Now it has grown to be better to distinguish false samples. Below the Tables 5.19 and 5.20 we can see that some words such as *GII . 4*, *influenxa A*, *PARV - A/B* represent a significant portion of false negatives. Looking at false positives we can see that there are still a lot of entities that end in punctuation marks, such as *West Nile (*, *. venerealis*, *WN) virus*. Those ones could be filtered

out by building some additional regular expression downstream features into the model.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	93.28 %	97.85 %	68.46 %	78.21 %
recall	86.00 %	89.59 %	77.84 %	88.79 %
F-score	89.49 %	93.54 %	72.85 %	83.16 %
False Positives	50 (37 only in rev)		275 (259 only in orig)	
False Negatives	113 (70 only in rev)		170 (140 only in orig)	

Table 5.17: Evaluation metrics - Part 1

	Difference	
	Exact	**Any overlap**
precision	**24.82 %point**	19.64 %point
recall	**8.16 %point**	0.8 %point
F-score	**16.64 %point**	10.38 %point
False Positives	-225	
False Negatives	-57	

Table 5.18: Evaluation metrics - Part 2

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
3 A virus	9 mice
2 WN	6 influenza A H1N1 (2009)
2 West Nile (	6 HIV
2 . venerealis	
1 WN) virus	
1 - Winged Sharpshooter	
1 S -	
Some samples only in revisioned	**Some samples only in original**
3 A virus	9 mice
2 West Nile (	6 influenza A H1N1 (2009)
2 . venerealis	6 HIV
1 WN) virus	
1 - Winged Sharpshooter	
1 S -	

Table 5.19: Some false positives with their frequencies S800 test.tsv evaluation run. See more detailed listing in appendix.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
10 influenza A	5 M2T2B15
6 alfalfa	5 influenza A H1N1
5 influenza A virus	
4 PERV - B	
4 influenza A viruses	
3 PERV - A	
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
10 influenza A	5 M2T2B15
5 broccoli	5 influenza A H1N1
4 PERV - B	
3 PERV - A	
1 porcine endogenous retrovirus (PERV) - B	

Table 5.20: Some false negatives with their frequencies S800 test.tsv evaluation run. See more detailed listing in appendix.

Train: LINNAEUS Test: LINNAEUS

There is quite a remarkable gap between the best devel model performance — around 97 % at the Table 5.1 — and the test set performance which is less than 86 % at the Table 5.21. This significant decrease raises questions about potential anomalies within the training, development, and test sets, supporting previous observations that the training and development sets may be overly similar to each other. One noticeable flaw is in false negatives, where around half of the scores stem from a file: *pmcA1838407.txt*. This file contains many instances of *Pileated Woodpecker, Ivory - billed Woodpeckerm Pileated Woodpeckers*. These terms and their derivations occur frequently in the Tables 5.23 and 5.24. It could perhaps be a reasonable idea to check the consistency of the annotation with the rest of the corpus. Filtering — a procedure to make the data set more consistent with the S800 — does make the performance worse as can be seen from the Table 5.22. This is expected as we have filtered out some of the most common names but there are no guidelines for all the common words.

	Original		**Filtered**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	87.84 %	92.85 %	87.34 %	91.39 %
recall	85.75 %	89.80 %	74.68 %	78.14 %
F-score	86.78 %	91.30 %	80.51 %	84.25 %
False Positives	170 (76 only in orig)		100 (6 only in filt)	
False Negatives	204 (22 only in orig)		234 (17 only in filt)	

Table 5.21: Evaluation metrics - Original and Filtered.

	Difference	
	Exact	**Any overlap**
precision	**0.5 %point**	1.46 %point
recall	**11.07 %point**	11.66 %point
F-score	**6.27 %point**	7.05 %point
False Positives	70	
False Negatives	-30	

Table 5.22: Evaluation metrics - Difference.

False Positives - Exact match	
Some samples from Original	**Some samples from Filtered**
20 Drosophila	20 Drosophila
16 Woodpecker	10 Vibrio cholerae
10 Vibrio cholerae	7 Photobacterium profundum
8 - billed	
6 Ivory - billed	
3 Woodpeckers	
Some samples only from Original	**Some samples only from Filtered**
16 Woodpecker	1 worm
8 - billed	1 Streptococcus Pneumoniae
6 Ivory - billed	1 S .
3 Woodpeckers	
1 woodpeckers	
1 woodpecker	

Table 5.23: linnaeus-corpus filtered/ - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from Original	**Some samples from filtered**
47 Pileated Woodpecker	47 Pileated Woodpecker
44 Ivory - billed Woodpecker	44 Ivory - billed Woodpecker
27 Pileated Woodpeckers	27 Pileated Woodpeckers
5 Ivory - billed Woodpeckers	5 Ivory - billed Woodpeckers

Table 5.24: linnaeus-corpus filtered/ - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

A closer look and a mathematical example of approximating evaluation performance in LINNAEUS

We can see that the vast majority of false negatives stem from Woodpecker-related words. Moreover, these words mostly came from article **pmca1838407**. It has the following words tagged entities: 47 Pileated Woodpecker, 44 Ivory-billed Woodpecker, 27 Pileated Woodpeckers and 5 Ivory-billed Woodpeckers, ... Thus in this article at least half of the FNs occur. This is very interesting and it would be very intriguing to know why this happens. However, it is out of the scope of this study. It is tempting to attribute poor performance of our model to flaws in the dataset. However, let's consider the possibility that this particular article was not selected for inclusion in the LINNAEUS dataset in the first instance. How would that affect the results? Discarding the other mentions in the article we can conclude that after there would be in total $204-47-44-27-5 = 81$ false negatives left (see the Table 5.25). Thus,

$$recall = \frac{TP}{TP + FN} = \frac{1228}{1228 + 81} \approx \ 93.88\% \tag{5.1}$$

$$F - score = 2 * \frac{precision * recall}{precision + recall} = 2 * \frac{87.84\% * 93.88\%}{87.84\% + 93.88\%} \approx 90.07\% \tag{5.2}$$

Which is roughly 4 percentage points up compared to the original 86.78 % (table 5.21). That is perhaps less than it would seem as the FNs decreased from 204 to 81, but it is certainly much better than 86.78 % and above the magic 90 % line. This shows that the corpus may be quite sensitive to a few but frequent mentions. If we extend our study to examine any overlap option rather than an exact match, we can see that it doesn't solve the issue. The model really fails to see even parts of the *woodpecker*-related entities.

To end this examination we can suggest that the problem with this particular

article may stem from naming conventions. We can see that the species tags follow the other: a common name followed by a scientific name. For example, the first entity ends at 280384 and the second start right after at 280385 etc (table 5.26). So perhaps that is the reason the model misses the entity.

Anyoverlap entities, which contain 'woodpecker'	
45	FN: Pileated Woodpecker
26	FN: Pileated Woodpeckers
26	FN: Ivory - billed Woodpecker
16	TP: Woodpecker
3	TP: Woodpeckers
2	FN: Ivory - billed Woodpeckers

Table 5.25: Woodpecker related terms account for many FNs

A peek at annotations of LINNAEUS	
T567	Species 280365 280384 Pileated Woodpecker
T568	Species 280385 280403 Dryocopus pileatus
T569	Species 280415 280440 Ivory - billed Woodpecker
T570	Species 280441 280464 Campephilus principalis
T571	Species 280556 280581 Ivory - billed Woodpecker
T572	Species 280582 280605 Campephilus principalis

Table 5.26: In 280385 species names tend to follow one after the other

5.5.2 Cross-corpus experiment

Train: S800 Test: LINNAEUS

It is quite intriguing to observe that the revision process has actually led to a deterioration in performance, as evidenced by the data presented in Tables 5.27 and 5.28. However, given that the difference in F-scores is less than one percentage point, this slight decline could potentially be attributed to random variations in model construction. Despite this, it's clear that the revisions have not enhanced performance. A plausible explanation for this phenomenon might be the misalignment between the LINNAEUS and the S800 datasets. Specifically, the inclusion of terms such as 'patients' and 'women' in LINNAEUS — which significantly contribute to the high rate of false negatives — makes the model's behavior unpredictable.

Regrettably, as evidenced by the subsequent experiment with the filtered LINNAEUS dataset, the pattern persists: the revision process has not resulted in improvement. Furthermore, as observed earlier, there is a significant discrepancy between the optimal model performance on the development set and its evaluation on the test set within LINNAEUS. This notable gap casts doubts on the overall integrity of the LINNAEUS dataset. Moreover, in both data sets, original and revisioned, recall is remarkably worse than precision. Some of that is due to the term *Pileated Woodpecker*, which stems from the document pmcA1838407. We did a small experiment, on how much removing that particular document affects the F-score if removed, and found out that it goes from 60.21 % to 61.83 %. So in this run, that document does not do a huge impact. The Table 5.29 shows that in both cases common terms 'patients' etc. along with *Pileated Woodpecker* are numerous in FN listing and in fact most of the FNs are the same in both original and revisioned as there are only 31 terms in revisioned FN listing that are not present in the original.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	86.31 %	92.83 %	81.79 %	85.66 %
recall	46.23 %	49.72 %	48.60 %	50.77 %
F-score	60.21 %	64.76 %	60.97 %	63.75 %
False Positives	105 (55 only in rev)		155 (102 only in orig)	
False Negatives	770 (31 only in rev)		736 (12 only in orig)	

Table 5.27: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	4.52 %point	7.17 %point
recall	-2.37 %point	-1.05 %point
F-score	-0.76 %point	1.01 %point
False Positives	-50	
False Negatives	34	

Table 5.28: Evaluation metrics - Part 2.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
151 patients	151 patients
74 women	74 women
65 men	65 men
46 Pileated Woodpecker	47 Pileated Woodpecker
46 patient	46 patient
36 persons	36 persons

Table 5.29: Train the S800 Evaluate on LINNAEUS - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: S800 Test: LINNAEUS filtered

As the previous run was evaluated against the original the LINNAEUS, this time we use the filtered version of the LINNAEUS for evaluation. The results are remarkably better as can be observed from the Tables 5.30 and 5.31. So clearly filtering has made the two data set more compatible. From the point of view of model comparison, it is also worth mentioning that the precision has gone up but the recall has gone down.

Moreover, even though the performance has gone worse in the exact match case, in the relaxed case of any overlap, it has gone up. That is true for both of the models the LINNAEUS and the LINNAEUS filtered. It is also worth noting that the number of false negatives is quite remarkable and a substantial amount of the results are from *Pileated Woodpecker* and *Pileated Woodpeckers*. This is interesting as it affects both of these models. If we take out that particular document, which contains all the references to the different Woodpecker mentions, namely the document pmcA1838407, we can see that the performance of the S800 revisioned goes from 77.86 % to 84.49 %, which is a remarkable difference. Underlining the effect that one randomly chosen document can make.

By removing pmcA1838407 from the original S800 data set, the F-score ends up at 82.73 %. This is an interesting result because as we shall see later when we use the LINNAEUS for testing, the original data set performs slightly better than the revisioned. However, if we remove that one document — which apparently causes a headache to all of our models — then it is the revisioned data set that does a better job.

It is also worth mentioning that the S800-rev has more false negatives than the S800-orig, and common terms such as 'calves', 'calf', 'cows', 'yeast' and 'chick' are frequent as can be seen from 5.32. Two last in the listing are only present in the S800-rev listing. Why are such many common terms in the listing? Does the S800

have more scientific names than the LINNAEUS and is thus prone to miss them?

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	85.40 %	92.89 %	81.30 %	85.13 %
recall	71.54 %	77.81 %	75.76 %	78.90 %
F-score	77.86 %	84.69 %	78.43 %	81.90 %
False Positives	113 (60 only in rev)		161 (112 only in orig)	
False Negatives	263 (43 only in rev)		224 (16 only in orig)	

Table 5.30: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	4.1 %point	7.76 %point
recall	-4.22 %point	-1.09 %point
F-score	-0.57 %point	2.79 %point
False Positives	-48	
False Negatives	39	

Table 5.31: Evaluation metrics - Part 2.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
47 Pileated Woodpecker	47 Pileated Woodpecker
35 Ivory - billed Woodpecker	27 Pileated Woodpeckers
27 Pileated Woodpeckers	25 calves
25 calves	19 calf
20 calf	16 Ivory - billed Woodpecker
11 cows	14 bovine
Some samples only in revisioned	**Some samples only in original**
8 yeast	6 human
6 Drosophila	2 V . cholerae
6 chick	2 mice

Table 5.32: Train the S800 Evaluate on the LINNAEUS filtered - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS Test: S800

Compared to the two previous runs, this run is the opposite. Now the LINNAEUS is used for training and the S800 for evaluation. Interestingly the pattern is quite different. Instead of the original version beating the revisioned one — a surprising trend which we saw earlier — the revisioned version is now taking the lead (see the Tables 5.31 and 5.34). Specifically, the number of false positives has decreased quite dramatically from 367 to 137 — take a look at the Table 5.33. This might seem quite odd since the training set has not changed, but because of that, we can conclude that the original S800 might have contained many untagged species tags, which were found by the tagger but were not annotated in the corpus.

Many of the false negatives found by the S800-orig seem quite gibberish. Terms such as 'YC6903' are frequent in the original version but seem to appear only in the original data set. Those terms — usually strains — have been left out from revisioned. Also, this time the terms *GII . 4* and *influenza A* are topping the FN listing on the revised side (see Table 5.35). So it seems that the same documents that decrease the F-score in-corpus (table 5.20) experiment are also present in cross-corpus experiments and hence there seems to be no easy solution with those FNs in question.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	82.59 %	88.06 %	60.15 %	71.38 %
recall	80.55 %	85.25 %	72.62 %	84.22 %
F-score	81.56 %	86.63 %	65.80 %	77.27 %
False Positives	137 (34 only in rev)		369 (245 only in orig)	
False Negatives	157 (58 only in rev)		210 (148 only in orig)	

Table 5.33: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	22.44 %point	16.68 %point
recall	7.93 %point	1.03 %point
F-score	15.76 %point	9.36 %point
False Positives	-232	
False Negatives	-53	

Table 5.34: Evaluation metrics - Part 2.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	8 YC6903
12 influenza A	6 M2 (T)
7 HAV	5 M2T2B15
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	8 YC6903
5 tobacco	6 M2 (T)
5 pneumococcal	5 M2T2B15

Table 5.35: Train on the S800 Evaluate on the LINNAEUS - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS filtered Test: S800

From the Tables 5.36 and 5.37, we observe a consistent trend: revised data outperforms the original. The filtering process effectively excludes false-positive terms like 'patient', observed in earlier sections. Moreover, the LINNAEUS shows an improvement from 81.56 % to 83.13 % when comparing filtered to original datasets, although the gain might not be as significant as expected. Interestingly, filtering benefits the original S800 version more, increasing from 65.80 % to 70.69 % , as shown in the Table 5.33. Notably, 'ant' and 'ants' emerge as prominent false positives in the filtered LINNAEUS dataset (Table 5.38), suggesting that common terms might be mistakenly associated with general terms like 'people', 'woman', 'women', 'patients', etc., rather than with specific scientific terminology.

The term 'GII.4' — present in Table 5.39 — predominates among the false positives (FPs) on the revised data, while entities in all caps, like 'YC6903', suggest strain or hinting abbreviations on the original dataset's side. This pattern of frequently occurring terms is consistent with earlier observations made with the unfiltered LINNAEUS dataset.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	85.47 %	91.10 %	67.62 %	77.86 %
recall	80.92 %	85.63 %	74.05 %	83.83 %
F-score	83.13 %	88.28 %	70.69 %	80.73 %
False Positives	111 (46 only in rev)		272 (186 only in orig)	
False Negatives	154 (59 only in rev)		199 (136 only in orig)	

Table 5.36: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	17.85 %point	13.24 %point
recall	6.87 %point	1.8 %point
F-score	12.44 %point	7.55 %point
False Positives	-161	
False Negatives	-45	

Table 5.37: Evaluation metrics - Part 2.

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
10 ant	14 mice
8 dengue	9 influenza A
7 ants	7 HIV
Some samples only in revisioned	**Some samples only in original**
10 ant	
7 ants	
3 fire ant	

Table 5.38: Train the LINNAEUS filtered Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
17 influenza A 15 GII . 4	8 YC6903 6 M2 (T)
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	8 YC6903 6 M2 (T)

Table 5.39: Train the LINNAEUS filtered Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

5.5.3 Combi-cross experiment

In the last section, we did cross-analysis between different data sets, using a data set from the S800 and a test set from the LINNAEUS and vice versa. In this section, we combine the training sets of the S800 and the LINNAEUS and evaluate them against the test sets from the S800 and the LINNAEUS.

Train: LINNAEUS and S800 Test: S800

The most interesting observation here is to see the comparison with the in-corpus run. We can see that by appending the LINNAEUS training set to the S800, the F-score of the S800 becomes better. We shall see, in the coming pages that this should not happen. Without the LINNAEUS the F-score of the revised S800 is 89.49 % (see Table 5.17) and the LINNAEUS appended it is 91.49 % (Table 5.40). 2 percentage points improvement is significant. The same trend holds for the original S800.

There is also a clear improvement of the S800-rev compared to the S800-orig in this study. The F-score has moved up by 17.71 % points (Table 5.41). Especially the number of false positives has fallen down remarkably. As we have discussed before *GII . 4* and *influenza* are hard cases and show up also in this case. They represent also this time a large proportion of FNs (Table 5.43).

In the realm of false positives, it's noteworthy that many common names, such as 'mice' and 'horses', frequently appear in the original dataset but are absent on the revised side, as shown in Table 5.42. Additionally, it's important to highlight that the most common false positives are unique to the original dataset and do not feature in the S800-revised listing. We will delve deeper into inconsistent terms later in the text. For instance, the inconsistency with 'mice' is illuminated by its classification: in the original dataset, 'mice' is labeled as a species nine times and not labeled nine times, whereas in the revised version, it is consistently marked as

a species 18 times. This underscores the importance of consistency.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	93.87 %	96.61 %	68.61 %	79.37 %
recall	89.22 %	91.45 %	79.79 %	91.00 %
F-score	91.49 %	93.96 %	73.78 %	84.79 %
False Positives	47 (22 only in rev)		280 (256 only in orig)	
False Negatives	87 (68 only in rev)		155 (141 only in orig)	

Table 5.40: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	25.26 %point	17.24 %point
recall	9.43 %point	0.45 %point
F-score	17.71 %point	9.17 %point
False Positives	-233	
False Negatives	-68	

Table 5.41: Evaluation metrics - Part 2.

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
3 WN	10 mice
2 Spruce	6 influenza A H1N1 (2009)
2 PERV	6 HIV
	5 horses
Some samples only in revisioned	**Some samples only in original**
2 PERV	10 mice
2 Neurospora	6 influenza A H1N1 (2009)
1 West Nile (	6 HIV
	5 pneumococcal
	5 horses

Table 5.42: Train the LINNAEUS and the S800 Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
9 influenza A	5 influenza A H1N1
6 alfalfa	5 FGSC
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
9 influenza A	5 influenza A H1N1
4 PERV - B	5 FGSC

Table 5.43: Train the LINNAEUS filtered and the S800 Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS filtered and S800 Test: S800

Filtering the LINNAEUS data set does not improve the performance when tested against the S800. In fact, the F-score is surprisingly a little bit down (see Tables 5.44 and 5.40). However, when regarding the F-scores of any overlaps instead of strict exact match, the model does slightly better: 94.52 % vs. 93.96 %. So one could argue that in this case filtering does not improve nor make things worse.

However, despite the lack of a significant difference between the filtered and non-filtered LINNAEUS datasets, there is a notable improvement in performance when comparing the original S800 (S800-orig) to its revised version (S800-rev) in combination with LINNAEUS. The F-score increases by 18.53 percentage points, as shown in Table 5.45. A plausible explanation might be that while terms like 'patients', 'men', and 'women' are often tagged as species in LINNAEUS, the model disregards them because they are not tagged in S800.

Could it be that an AND logic operates in the learning process? If the same terms are tagged as species in some instances but not in others, do they negate each other? Testing this hypothesis would be straightforward with a purposely crafted artificial dataset.

We can see that also this time the S800-rev suffers from lower recall than precision and at least the most frequent FN terms (Table 5.46) *GII . 4*, *influenza A* and *PERV A/B* are the ones which are problematic in other runs as well.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	92.70 %	96.41 %	69.14 %	79.39 %
recall	89.71 %	92.69 %	76.53 %	87.35 %
F-score	91.18 %	94.52 %	72.65 %	83.18 %
False Positives	57 (27 only in rev)		262 (232 only in orig)	
False Negatives	83 (56 only in rev)		180 (159 only in orig)	

Table 5.44: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	23.56 %point	17.02 %point
recall	13.18 %point	5.34 %point
F-score	18.53 %point	11.34 %point
False Positives	-205	
False Negatives	-97	

Table 5.45: Evaluation metrics - Part 2.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
11 influenza A	5 tobacco
4 PERV - B	5 6C (T)
3 PERV - A	4 LPU83
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
11 influenza A	5 M2T2B15
4 PERV - B	5 influenza A H1N1
3 PERV - A	5 FGSC

Table 5.46: Train the LINNAEUS filtered and the S800 Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS and S800 Test: LINNAEUS

Contrary to previous experiments, in which the LINNAEUS and the S800 were used together against the S800, this time they are tested against the LINNAEUS. We can see that this time there is no improvement but in fact disimprovement, between the original and revised versions of the S800. This is quite a remarkable fact since in the previous section the improvement was nearly 18 % points — see the Table 5.41 — and now it has decreased by nearly -2 % points (Table 5.48).

Precision is down around half a pp but recall almost 3 pp in favour of the S800-orig. As we look at the FNs on the Table 5.49, we can see that the previously discussed 'woodpecker' document (pmca1838407) explains around half of the FNs. Still, there are around 40 more FNs on the S800-rev side and as we can see from the listing there seems not to be any particular term that could explain the difference. The term *Vibrio MED222* has 3 hits that appear only in rev but all the rest have only 2 or 1. So the gap is evenly distributed and suggests that the S800-rev is really worse this time. The FP listing does not seem to reveal anything remarkable. The 'woodpecker' document also shows up to some extent but the term *Drosphila* is frequent in both the S800-rev and the S800-orig. (see details in appendix A.30)

Upon closer examination of the listing on the Table 5.49, we notice that many false negatives (FNs) conclude with uppercase abbreviations (*YJ016, V52...*), suggesting that the model does not recognize these terms as species, though they indeed are. Given their exclusive presence in S800-rev, it's logical to infer that these are errors in LINNAEUS arising from the revision rules, specifically the guideline to exclude strain names from tagging. Therefore, while S800-rev may underperform compared to S800-orig in this context, it arguably demonstrates a valuable learning outcome: the importance of excluding strain names.

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	90.74 %	94.87 %	91.28 %	94.33 %
recall	82.75 %	86.24 %	85.47 %	88.06 %
F-score	86.56 %	90.35 %	88.28 %	91.09 %
False Positives	57 (27 only in rev)		262 (232 only in orig)	
False Negatives	83 (56 only in rev)		180 (159 only in orig)	

Table 5.47: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	-0.54 %point	0.54 %point
recall	-2.72 %point	-1.82 %point
F-score	-1.72 %point	-0.74 %point
False Positives	-205	
False Negatives	-97	

Table 5.48: Evaluation metrics - Part 2.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
47 Pileated Woodpecker	45 Pileated Woodpecker
42 Ivory - billed Woodpecker	27 Pileated Woodpeckers
27 Pileated Woodpeckers	20 calves
21 calves	19 Ivory - billed Woodpecker
16 calf	18 calf
10 chick	10 chick
Some samples only in revisioned	**Some samples only in original**
3 Vibrio MED222	2 V . cholerae
2 zebrafish	2 bovine
2 women	1 man
2 V . vulnificus YJ016	1 Lutzomyia whitmani
2 Vibrio cholerae V52	1 Lutzomyia neivai
2 Vibrio cholerae V51	1 Lutzomyia intermedia

Table 5.49: Train on the LINNAEUS and the S800 Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS filtered and S800 Test: LINNAEUS filtered

In the last run, we used the LINNAEUS and the S800 against the LINNAEUS. In this run, we use the same setup but use the filtered version of LINNAEUS, which should be more in line with the S800-rev. However as we can see with both cases — the S800-rev and the S800-orig — the F-score numbers are down by around 5 - 10 pp (Tables 5.47 vs. 5.50). So filtering common terms makes things worse. This trend is in line with the in-corpus study, in which the F-score of the filtered version was around 5 pp lower than the non-filtered version, settling to 80.51 % (Table 5.21). So adding the S800 to the LINNAEUS makes the data set perform worse than without the addition — 80.19 % (Table 5.50) vs. 80.51 % (Table 5.21).

As regards to differences between rev. and orig. data sets, now the favour is on the side of the revisioned corpus. Even though the winning margin is not remarkable, only around 1.3 % points (Table 5.51). This is interesting as in other studies, in which the evaluation is against the LINNAEUS, surprisingly the S800-orig is better than the S800-rev. In this instance, however, the revised version proves to be superior. Additionally, an intriguing observation from both this and the previous study is the minimal presence of common terms like 'patients', 'men', and 'woman' in the list of false positives (FPs). Thus, while filtering significantly impacts certain studies, such as training on S800 and evaluating against LINNAEUS (Table 5.27), in scenarios where LINNAEUS and S800 datasets are combined, common terms seldom appear in FP listings.

More detailed FN and FP analysis reveal that FN terms are once again dominated by the 'Woodpecker' document. The related terms make up half of the FNs. In the FP-listing (see appendix A.31) there seems to be no particular pattern. One peculiar case is with the term *Drosophila* with 19 mentions in orig. and 10 in rev. This term is hard, as the writer uses Genus name in place of the full species name *Drosophila melanogaster* [16].

	Revisioned		**Original**	
	Exact match	**Any overlap**	**Exact match**	**Any overlap**
precision	87.47 %	93.09 %	85.09 %	91.35 %
recall	74.03 %	78.57 %	73.48 %	78.35 %
F-score	80.19 %	85.22 %	78.86 %	84.36 %
False Positives	98 (22 only in rev)		119 (36 only in orig)	
False Negatives	240 (9 only in rev)		245 (29 only in orig)	

Table 5.50: Evaluation metrics - Part 1.

	Difference	
	Exact	**Any overlap**
precision	2.38 %point	1.74 %point
recall	0.55 %point	0.22 %point
F-score	1.33 %point	0.86 %point
False Positives	-21	
False Negatives	-5	

Table 5.51: Evaluation metrics - Part 2.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
47 Pileated Woodpecker	47 Pileated Woodpecker
43 Ivory - billed Woodpecker	39 Ivory - billed Woodpecker
27 Pileated Woodpeckers	27 Pileated Woodpeckers

Table 5.52: Train the LINNAEUS filtered and the S800 Evaluate on the S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

5.6 Analyzing document level F-scores

As we can see, some of the words seem to make up a large proportion of false negatives. We shall take a closer look what is an effect of a single document on the F-score. We use two metrics, to try to capture this effect. One solution would be to take the proportion of a species in a document and divide it by its F1 score. — equation 5.3. Hence the more species one document contains, the more it affects this number and reversely the smaller the F1-score is the higher will be the number.

$$\text{F1-IMP1}_{\text{doc}} = \frac{\text{Species-proportion}}{\text{F1-score}_{\text{doc}}} \tag{5.3}$$

The downside of this method is that it also marks documents in which the F-score is high if the proportion of species is high enough. Because of that, we use mainly the following formula (5.4) to analyze the negative effect of a document on an overall F-score.

$$\text{F1-IMP2}_{\text{doc}} = (100\% - \text{F1-score}_{\text{doc}}) \times \text{Species-proportion} \tag{5.4}$$

5.6.1 LINNAEUS

In the Table 5.53 and in the Figure 5.4 we have gathered all the documents and some of their features regarding species tags. There are two documents that stand out with the formula 5.4, which we refer to as F1 imp. 2 in the Table. These two documents are pmcA1838407 and pmcA1885853 (marked in the Table 5.53) and stand out in the Figure 5.4. The first one is especially significant according to metric, it has a score over 15 while other documents settled close to zero. In the Table 5.54, the annotations for the document are detailed. We can see, that the document mentions the term 'Woodpecker' in various forms. We can see these terms appear over and over again in the following model studies and in most cases

they make up around half of the false negatives, so this particular document is an issue in our study.

Table 5.53: Document-level features and their F-scores of the LINNAEUS data set

DOCID pmcA-	Lines %	Species %	Spec. dens.	Precision	Recall	F1	F1 imp.1	F1 imp.2
1036069	2.79	0.89	0.43	100.00	50.00	66.67	1.33	0.30
1075922	4.76	1.29	0.37	100.00	92.31	96.00	1.34	0.05
1079799	1.91	0.31	0.22	100.00	100.00	100.00	0.31	0.00
1257442	2.89	2.82	1.32	100.00	100.00	100.00	2.82	0.00
1347483	1.88	0.17	0.12	66.67	100.00	80.00	0.21	0.03
140144	2.82	5.6	2.69	100.00	100.00	100.00	5.60	0.00
1562423	1.75	1.83	1.42	100.00	100.00	100.00	1.83	0.00
1634875	2.78	13.44	6.54	85.95	100.00	92.44	14.54	1.02
1779435	1.43	2.37	2.24	100.00	100.00	100.00	2.37	0.00
1779441	4.74	3.04	0.87	98.55	100.00	99.27	3.06	0.02
1805739	3.88	2.5	0.87	86.44	91.07	88.70	2.82	0.28
*1838407	3.03	16.39	7.33	15.22	5.26	7.82	209.59	15.11
*1885853	3.68	6.89	2.54	40.00	42.31	41.12	16.76	4.06
1891629	1.1	4.48	5.51	67.27	97.37	79.57	5.63	0.92
1931593	4.89	2.15	0.59	100.00	100.00	100.00	2.15	0.00
2151861	0.74	0.53	0.97	85.71	100.00	92.31	0.57	0.04
2365090	4.74	0.62	0.17	81.25	92.86	86.67	0.72	0.08
2365225	1.9	0	0	0.00	0.00	0.00	0.00	0.00
2398786	4.19	11.78	3.81	97.25	100.00	98.60	11.95	0.16
2412862	4.78	3.27	0.92	98.59	98.59	98.59	3.32	0.05

2531092	0.57	0.13	0.31	75.00	100.00	85.71	0.15	0.02
2547183	2.87	0	0	0.00	0.00	0.00	0.00	0.00
2553092	2.98	0.58	0.26	90.00	100.00	94.74	0.61	0.03
2556924	5.31	3.13	0.8	97.01	92.86	94.89	3.30	0.16
2567314	2.46	2.82	1.55	83.82	90.48	87.02	3.24	0.37
2599109	2.58	0.98	0.51	100.00	95.45	97.67	1.00	0.02
2636784	5.84	1.38	0.32	60.53	79.31	68.66	2.01	0.43
2636797	4.04	3.04	1.02	95.71	98.53	97.10	3.13	0.09
280693	3.22	2.41	1.02	100.00	81.40	89.74	2.69	0.25
328326	3.71	0.26	0.09	40.00	66.67	50.00	0.52	0.13
328328	2.46	1.43	0.79	86.11	100.00	92.54	1.55	0.11
335194	1.07	1.7	2.15	93.75	100.00	96.77	1.76	0.05
509286	2.03	1.61	1.07	100.00	100.00	100.00	1.61	0.00

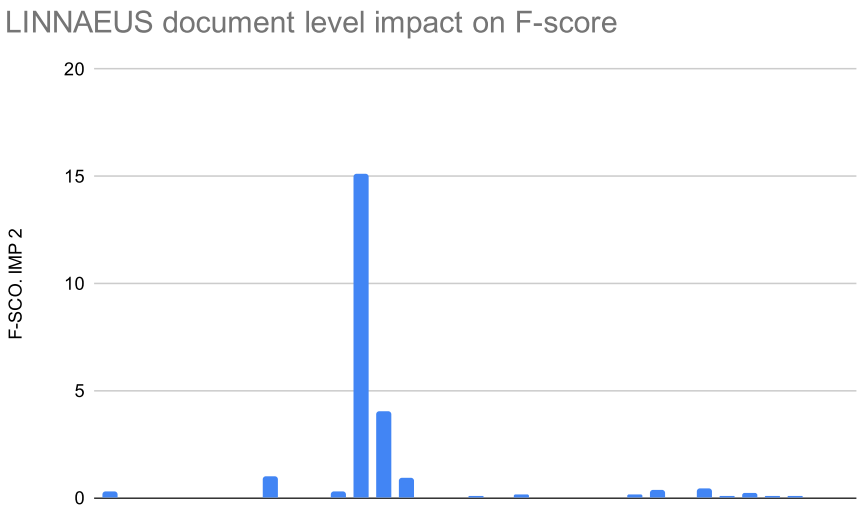

Figure 5.4: Examining the effect of a single document on F-score

The other document that stands out is pmcA1885853, which contains a lot of words with capitalized abbreviations such as *CMCP6, SS9, MED222* etc. — listed in 5.55. These instances show up also in the false-negative listing in the Table A.16.

47	Pileated Woodpecker
44	Ivory - billed Woodpecker
27	Pileated Woodpeckers
5	Ivory - billed Woodpeckers
3	Dryocopus pileatus
3	Campephilus principalis
1	White - tailed Deer
1	Pileated woodpeckers
1	Pileated
1	Ivory - bill

Table 5.54: Annotation in pmcA1838407 gold

If we look at how well the tagger does when we relax exact match to any overlap we can see, that the F-score goes to 95.33 %. So in this case it is most likely a problem with the abbreviations.

Effect of pmcA1838407 to the F1-score of the LINNAEUS

Extending the study we did in *In-corpus experiment* section and specifically in the equation 5.2, which estimates effects on the F1-score. Here we have taken into account all the factors of leaving out a single document — not just false negatives but also all the true and false positives — and did a new test run, in which the pmcA1838407 (the woodpecker document) was completely removed from the LINNAEUS corpus, and found out that indeed it has a significant effect. As we can see from the Table 5.56.

This is a very significant improvement, from 86.78 % to 92.12 %. So leaving out around 3 % of the lines from the test-set, makes the model from a moderate performance model, to a high-level model. This doesn't mean that when a model fails, it's always the fault of the data; rather, chance can play a crucial role in

7	V. cholerae
4	V. vulnificus CMCP6
4	Photobacterium profundum SS9
3	Vibrio MED222
3	Photobacterium profundum 3TCK
2	V. vulnificus YJ016
2	Vibrio cholerae V52
2	Vibrio cholerae V51
2	Vibrio cholerae RC385
2	Vibrio cholerae MO10
2	Vibrio cholerae 0395
2	Shewanella oneidensis
...	

Table 5.55: Annotation in pmcA1885853 gold

precision	90.31 %	(1221/1352)
recall	94.00 %	(1221/1299)
F-score	92.12 %	

Table 5.56: Test data set without the document pmcA1838407

determining whether we deem a model to be good or not. For example for the LINNEAUS 160 documents were selected randomly. If one of the documents had been left out, we would consider the species NER problem nearly solved, but with the document in, that is not the case. This is also applicable to other similar issues. We might have trained a nearly perfect model for a specific task, but perhaps we were just lucky. There could have been a document that, if selected, could have

45	L . whitmani
42	L . intermedia
6	Drosophila
5	sand flies
...	

Table 5.57: pmcA1634875 gold annotations. The majority of them belong to two classes.

significantly worsened the outcome.

So a better metric than just the F-score of the full document would be to do a document-level F-score analysis and then look for outliers. In this case, a relevant question is why the woodpecker file fails so badly. One remark is that it has a very high density of species among the text, over 7 % of the content of the document is tagged as species. This alone seems not to be the problem, as there is also another document pmcA1634875, which has nearly as high as species density: 6.54 %. It however does well in terms of the F-score: 92.44 %. In the Table 5.57 we take a peek at the top annotations of pmcA1634875 and can see, that L. withmani and L. intermedia make up the majority of the tags. Seemingly the model understands them well.

Let us take a look at predictions versus gold annotations in this file. Some of them are presented in the Table 5.59.[1] That Table shows that there is no repeating pattern explaining what could go wrong, but on the contrary, all the possibilities are more or less covered. Because the BERT attention mechanism looks at the surroundings of a particular word, we should present them also and not only the species words.

However, the Table 5.59 offers one perspective: The species in this document are

[1]Side note: It is nice to see that the BERT model indeed does look at the context and not just words. The term Ivory-Billed Woodpecker is treated very differently throughout the file.

particularly lengthy. And indeed if we make a study, we can see that this particular document, pmcA1838407, has most of the longest words in the corpus. It has 50 words that span 4 lines in the tsv-file. This negative effect is somewhat obvious as it is reasonable that the longer the tag, the harder it is to find it.

Beyond its length, the document exhibits a notable feature related to encoding: the frequent use of hyphens, exemplified by names like 'Ivory-billed Woodpecker'. Additionally, hyphens are found exclusively within the species names in both the training and testing datasets, as detailed in the Table 5.58. Hyphen has also other uses in language, and that might affect the results. How to test the effect of the hyphen: A reasonable scenario would be to put words with hyphens also in the devel set and see how it affects the performances, as now the devel set performs extremely well.

Table 5.59: Comparison between test and gold run of the pmcA1838407 run :

In some cases the term Woodpecker is marked individually			
Ivory	O	Ivory	B-Species
-	O	-	I-Species
billed	O	billed	I-Species
Woodpecker	B-Species	Woodpecker	I-Species
Campephilus	B-Species	Campephilus	B-Species
principalis	I-Species	principalis	I-Species
Some cases only latin name is marked			
Pileated	O	Pileated	B-Species
Woodpecker	O	Woodpecker	I-Species

Dryocopus	B-Species	Dryocopus	B-Species
pileatus	I-Species	pileatus	I-Species

In many occasions the term is missed completely

woodpeckers	B-Species	woodpeckers	O

Also with the longer form

Pileated	O	Pileated	B-Species
Woodpeckers	O	Woodpeckers	I-Species

And with the longest version

Ivory	O	Ivory	B-Species
-	O	-	I-Species
billed	O	billed	I-Species
Woodpecker	O	Woodpecker	I-Species

And sometimes only Pileated is missed

Pileated	O	Pileated	B-Species
Woodpecker	B-Species	Woodpecker	I-Species

Sometimes - is marked as I-Species and there is no B-Species before them

Ivory	O	Ivory	B-Species
-	I-Species	-	I-Species
billed	I-Species	billed	I-Species
Woodpecker	O	Woodpecker	I-Species

Woodpecker is understood as an individual unit

Ivory	B-Species	Ivory	B-Species
-	I-Species	-	I-Species
billed	I-Species	billed	I-Species
Woodpecker	B-Species	Woodpecker	I-Species

Test set	
1	Ivory-bill
44	Ivory-billed Woodpecker
5	Ivory-billed Woodpeckers
1	White-tailed Deer

Devel set
no mentions

Train set	
2	foot-and-mouth disease virus
11	guinea-pig
18	guinea-pigs
3	Guinea-pigs
3	HIV-1
1	Sprague-Dawley rats
1	Synechococcus strain Tx-20
1	Synechococcus TX-20

Table 5.58: Hyphen inside species tags in LINNAEUS

5.7 S800

As there are much more documents in the S800 than there are in the LINNAEUS, we won't show the statics for them all. In the Table 5.60 are the first and last documents with the ones that stand in the impact score. We can see that the F1-impact score has decreased in many cases but also increased in others. One notable increase is the document 20980508, which has gone from 0.245 to around 1.9. In the Figure 5.6 documents of the S800 are mapped with respect to their old impact score (on X-axis) and their new impact score (on Y-axis). The Y-axis shows smaller

scores altogether, but two documents pop out as outliers. They are 20933542 and 20980508. F1-impact scores are also plotted as document ID vs impact score on Figure 5.5. We can see that the plot shows that there are no extreme outliers that were present in LINNAUES (see the Figure 5.4). That is partly due to the fact that there are 800 documents in the S800 whereas only 160 are in LINNAEUS, but all in all, the revisioning process has made many of the peaks disappear.

DOCID	Original species	Revision species	Original impact	Revisioned impact
20139281	27	12	0.638614	0.06363
20139284	15	12	0.615	0.05383
20173003	37	8	1.748484	0.051106
20173005	64	12	3.309375	0
20173009	44	19	1.295919	0
...	...	...	...	...
20933542	22	38	0.605	1.47
...	...	...	...	...
20980508	9	57	0.245	1.897048
...	...	...	...	...
21029748	91	17	3.400548	0.770022
...	...	...	...	...
21906194	14	14	0	0
22182604	10	10	0	0
22182607	29	23	0.35552	0.23651
22182609	11	12	0.23076	0
22182616	26	30	0.376376	0.6363

Table 5.60: F1 impact on the S800-original and the S800-revision. F1-impact has diminished in most cases, but not all

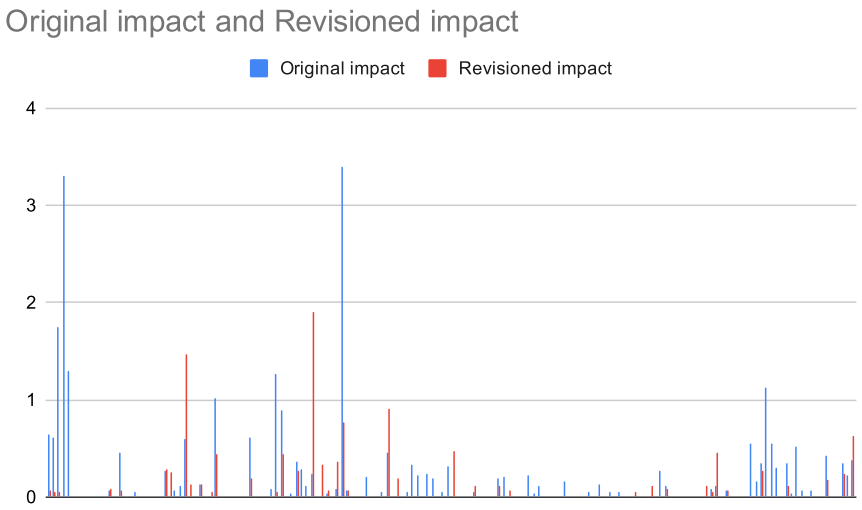

Figure 5.5: Document on X-axis vs their F1 impact score

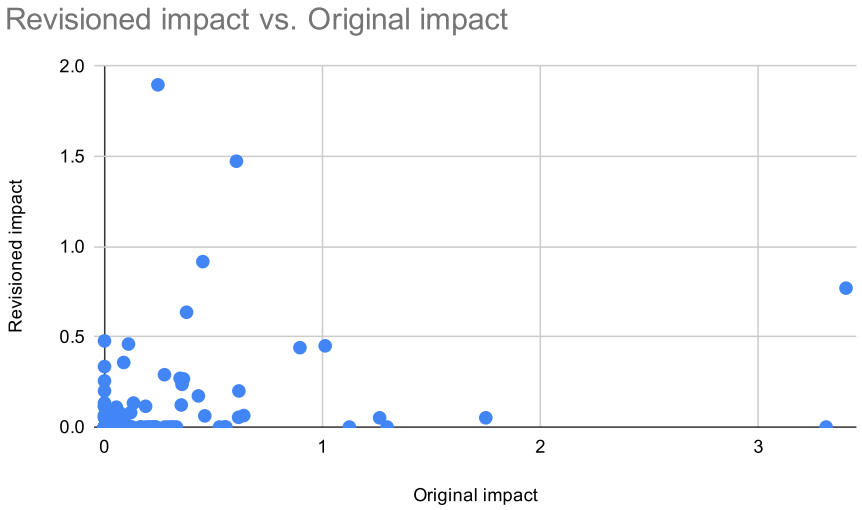

Figure 5.6: Examining the effect of a single document on F-score

Based on this analysis, it's evident that two documents in the revised version notably diverge from the others: 20980508 and 20933542. The first appears to be an error, containing 15 mentions of GII.4 — a strain that is documented in Table 5.61. That high number of missed entities shows throughout the whole study in many of the combined data sets. GII . 4 does not appear as species in the S800-orig, so it must be a new annotation error. The other document with a high F1 score impact is assigned to 20933542, that document contains lots of species of *Porcine endogenous retroviruses* known as PERV. That entity will also be in the false positive listings of many combined models. The Table 5.62 shows the annotations of this file, and we can see that once again the problem seems to reside in strains, such as PREV - A and PREV - B strains. The model, arguably making a prudent decision, retains the 'A and B' suffixes. However, likely due to an error in the annotations, it subsequently errs. Both Tables 5.61 and 5.62 demonstrate that while the model successfully identifies the straightforward cases, it fails with those that contain strain suffixes.

Let us take a look at one more file 21029748, which fails miserably in the original model but also stands out in bad light in the revisioned study. Table 5.64 shows the original annotations, which we can see contains very long entities that can span up to 12 lines in the tsv file as *Pandemic influenza A / H1N1 2009 (A / H1N1pdm) virus*, no wonder this is a hard case for any model. Table 5.63 shows how stripped this file has become after strains have been left out and actually it is still not easy as the suffix A makes easily think that it is a strain. The term influenza A shows up in the test studies of various models. We can see that by looking and comparing the top entities in at A.14 and at the three listings: 5.62, 5.61 and 5.63, we can see that indeed they have much in common. So the F1-score impact method we introduced in 5.4, seems to catch something essential at least in this study, and a similar method could be used in other studies to spot where the model fails. This is important as we saw with the pmcA1838407 that a single document can make

overall performance turn from excellent to moderate.

Entities in the S800-rev gold set	
15	GII . 4
2	mice
2	human
1	Norovirus GII . 4
1	HIV
1	hepatitis C virus

Entities in the S800-rev predictions set	
2	mice
2	human
1	HIV
1	hepatitis C virus

Table 5.61: 20980508 shows many instances of GII . 4, which is a strain and therefore a mistake in the data set

Entities in the S800-orig gold set	
2	influenza viruses
2	A / H1N1pdm virus
1	viral strains of A / H1N1pdm
1	seasonal influenza A / H3N2 viruses
1	seasonal influenza A / H1N1
1	seasonal A / H1N1
1	Pandemic influenza A / H1N1 2009 (A / H1N1pdm) virus
1	pandemic influenza A / H1N1 2009
1	influenza A viruses

1	human
1	A / Sakai / 89 / 2009 (H1N1) pdm
1	A / H3N2 influenza viruses
1	A / H3
1	A / H1pdm
1	A / H1
1	A / Aichi / 472 / 2009 (H1N1) pdm

Entities in the S800-orig predictions set

2	A /
1	seasonal influenza A / H3N2 viruses
1	seasonal influenza A / H1N1
1	pandemic influenza A / H1N1 2009
1	influenza A viruses
1	human seasonal A / H1N1
1	H1N1
1	A / Sakai / 89 / 2009 (H1N1)
1	A / H3N2 influenza viruses
1	A / H1N1 2009
1	A / Aichi / 472 / 2009
1	/

Table 5.64: 21029748 shows an example of long strain names in the S800-orig. The model picks them out surprisingly well

Entities in the S800-rev gold set	
4	PERV - B
3	PERV - A
2	PERVs
1	porcine endogenous retrovirus (PERV) - B
1	Porcine endogenous retroviruses
1	pigs
1	PERV
1	murine
1	human

Entities in the S800-rev predictions set	
2	PERV
1	porcine endogenous retrovirus
1	Porcine
1	pigs
1	PERV -
1	murine
1	MoMLV
1	human

Table 5.62: Species annotations in the 20933542 document in the S800-rev.

Entities in the S800-rev gold set	
4	influenza A
1	influenza A viruses
1	human
1	A / H3N2 influenza viruses

Entities in the S800-rev predictions set	
1	influenza A
1	human

Table 5.63: 21029748 document shows influenza A instances in the S800-rev

5.8 Analyze by category

In the following section, we conduct comparisons between the BERT-based NER system and the original SPECIES tagger. It's important to recall that the objective of the initial study was not only to identify entities but also to normalize them. Our focus extends beyond mere localization. It could be argued that exact matches yield the complete species name, which might then be utilized with a predefined dictionary for downstream normalization to a species ID. However, this approach has limited applicability since, in numerous instances, the specific species in question remains ambiguous, as demonstrated by the C. Sativa example in the SPECIES study.

The original paper of species shows a Figure, which shows precision and recall of the eight categories: *virology, botany, botany, zoology, bacteriology, medicine, mycology, entomology*, and *protistology* — see Figure 5.7.

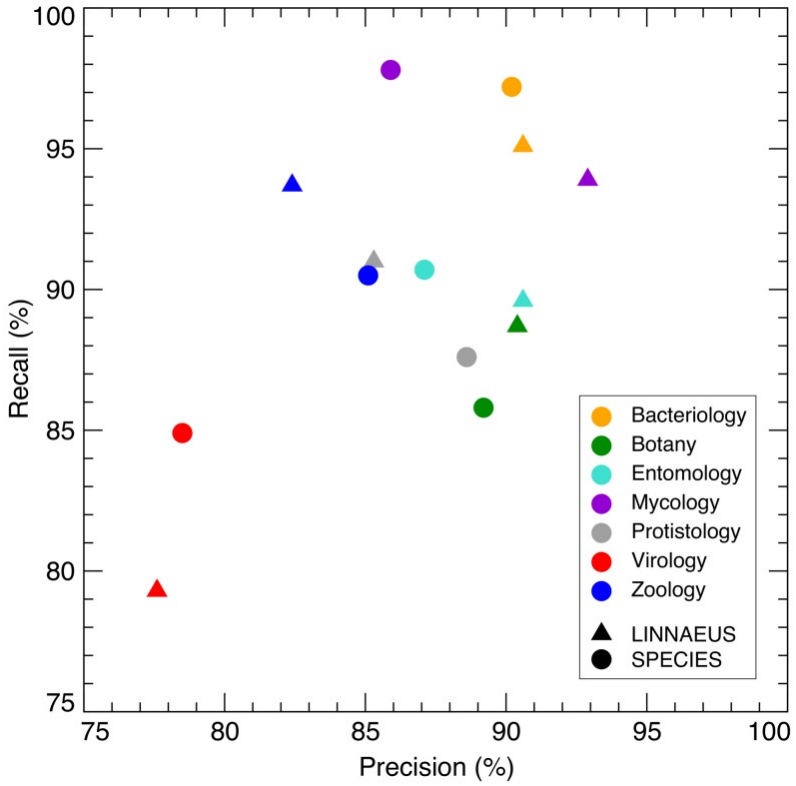

Figure 5.7: Diagram showing original results precision vs. recall from the original SPECIES paper. Source: https://www.researchgate.net/figure/Because-the-S800-corpus-consists-of-seven-different-taxonomic-categories-the-eighth_fig2_245027453 License CC 4.0

A direct comparison of our model to the one discussed in the paper isn't feasible since the study therein utilizes the complete dataset of 100 abstracts from each category. Conversely, our test set is stratified, with each category containing 20 documents, totaling 160 documents. In the Table 5.66, we've extrapolated the performance metrics for each category, with the SPECIES column reflecting the scores from the original SPECIES study. Here, we compare the F-scores—a harmonic mean of precision and recall—between the original SPECIES model and our revised NER-BERT model developed for S800. Similarly, the Table 5.65 facilitates a comparison between the S800 original and our revised models.

Category	**Orig. F-score**	**Rev. F-score**	**Change pp**	**Change %**
bac	78.65%	87.72%	9.07%	12%
bot	93.81%	84.96%	-8.85%	-9%
ent	76.45%	93.38%	16.93%	22%
med	56.67%	85.71%	29.05%	51%
myc	82.40%	96.04%	13.64%	17%
pro	45.71%	94.69%	48.98%	107%
vir	72.37%	72.49%	0.12%	0%
zoo	81.60%	85.71%	4.11%	5%
AVG	**73.46%**	**87.59%**	**14.13 pp.**	**26%**

Table 5.65: S800-orig vs. S800-rev F-score change by category

Category	**SPECIES F-score**	**Rev. F-score**	**Change pp**	**Change %**
bac	93.62%	87.72%	-5.90%	-6%
bot	87.47%	84.96%	-2.51%	-3%
ent	88.84%	93.38%	4.55%	5%
med	NODATA	85.71%	NODATA	NODATA
myc	91.61%	96.04%	4.43%	5%
pro	88.00%	94.69%	6.69%	8%
vir	81.08%	72.49%	-8.58%	-11%
zoo	87.66%	85.71%	-1.95%	-2%
AVG	**88.32%**	**87.59%**	**-0.74 pp.**	**-1%**

Table 5.66: SPECIES vs. S800-rev F-score change by category

First of all, we can see that S800-rev has done huge improvements in protistology and medicine, and minor improvements in bacteriology, entomology, mycology and zoology. No improvement in virology and negative improvement in botany. Intriguingly, botany performed best under the original model but, excluding virology,

it ranks as the least improved area in the revised model. This is quite surprising as one could expect botanical names to be quite standardized in their nomenclature. Another interesting remark is that virology has not improved practically at all. These findings strongly suggest that further studies should be done to improve these categories. See also Figures 5.8 and 5.9.

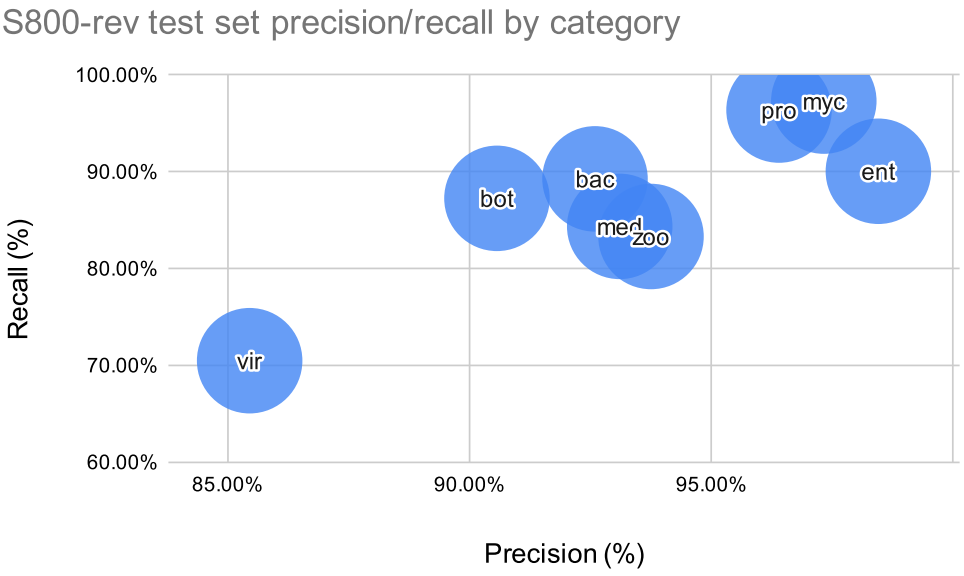

Figure 5.8: S800 revision test data set precision vs. recall by category

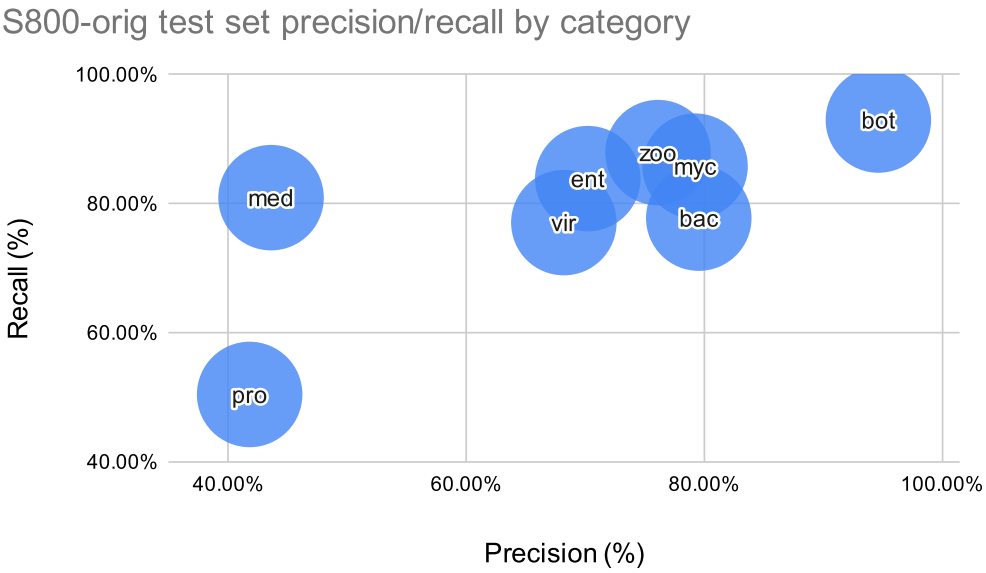

Figure 5.9: S800 original test data set precision vs. recall by category

When comparing the two methods - developing a NER tagger using the general-purpose language model BERT, and the rule-dictionary-based tagger such as the SPECIES - we observe minimal overall difference. There's a slight decrease in performance for bacteriology, botany, and zoology, some improvements in protistology, mycology, and entomology, and a notably significant decrease in virology. We suggest that the NER-based model offers a more practical solution for creating the Species-NER-tagger, given that it was developed solely with the S800 dataset from the SPECIES project, which was annotated specifically for testing. However this argument should not be taken too seriously as our NER-tagger does not normalize species mentions, but merely finds them.

5.8.1 Rate on learning

An intriguing aspect to explore is the impact of training set size on the learning curve. We created several datasets, gradually increasing the number of documents allocated for training, with this augmentation executed through careful stratification by cat-

egory. Our experiments included analyses on the complete dataset—illustrated in Figure 5.10—and another dataset from which all virology documents were excluded, as shown in Figure 5.11. The stratification ensured a proportional representation of articles from each category at each increment stage. We can see, that in fact, the model does not improve significantly after 50 % of documents are added. Interestingly only a tenth of articles used for training provides quite good results. This suggests that getting better results in finding species, should not be done purely by increasing the training data set size, but by trying to focus on the most hard cases. One option could be to decouple finding viruses from the other species. This however may not yield good results, as papers that deal with virology, also contain other species names. As Table 5.11 suggest the same trend holds for the data set, that contains no viruses. Notice that we have left all virology-related items also from the test set and devel set. As we can see the results are much better, and also have the same trend: No additional training data is needed after around 50 % of training data. In this light, the process of extending S800 to contain 1000 documents, S1000, is not perhaps the right direction.

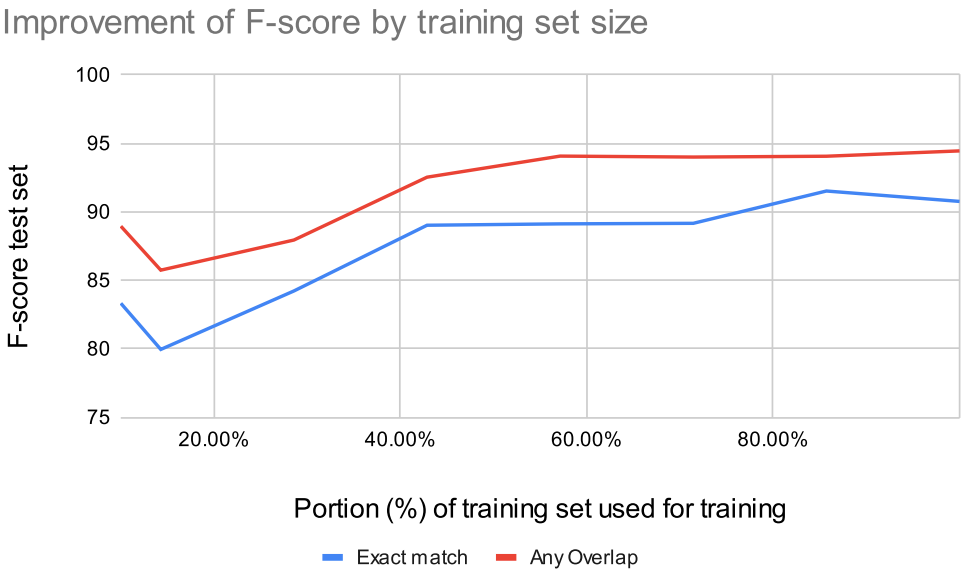

Figure 5.10: F-score of the test set by increasing training. Stratified on all categories.

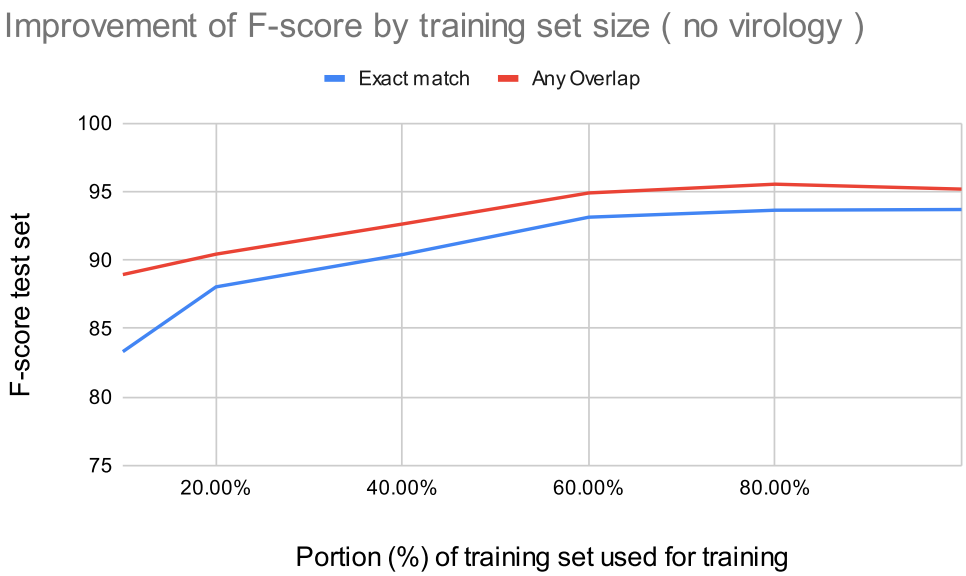

Figure 5.11: F-score of the test set by increasing training. Stratified on all categories.

6 Conclusions

In this thesis, we focus on the enhancement and redefinition of the S800 dataset. At the beginning, we presented the following research questions that we aim to answer:

1. How much improvement re-evaluation of the S800 dataset gives? We do this on in-coprus, cross-corpus and combined-cross-corpus corpora.
2. How much training set size contribute to performance. Is the model getting better with more training data?
3. Are there categories of species which are particularly difficult to recognise?
4. Is looking merely at overall F-score reasonable and what can be found by more extensive evaluation strategies?

In this final section, we summarize the results and revisit these questions.

Development Results

Development results were obtained by conducting grid runs across various dataset combinations, and from which the optimal model was identified. The Table 6.1 summarizes the hyperparameter optimization results for the development data and shows that the revised versions significantly outperformed the original in all categories.

The performance enhancement between the original and revised versions ranged from 10 to 21 points in evaluations against the S800 and from 1 to 9 points against

various versions of the LINNAEUS, indicating consistent improvements throughout the development set.

Test Results

Test runs were evaluated against the best performing model from the deveploment runs. To give answer to the first reseach question: **The final F1-score for the in-corpus S800 improved significantly from 72.85 % to 89.49 %** — see the Table 6.1 — marking a performance increase of over 22 %. Across all models tested with S800, the revised versions consistently outperformed the original, with improvements ranging from 20 % to 25 %. However, results were less notable when using LINNAEUS or LINNAEUS filtered data, with negligible or slightly better performance in favor of the original dataset. This suggests limited enhancements against the LINNAEUS test set. Remarkably, reversing the roles—evaluating LINNAEUS against the revised S800—yielded nearly a 25 % boost in performance.

Difference between Test and Development

In Table 6.2 we have analyzed the performance differences between development and test runs. For instance, the calculation in the upper left cell of the Table 6.2 is $\frac{74.49-72.85}{72.85} = 2.25\%$, where positive values indicate superior development set performance, and negative values indicate better test set performance.

Evaluation data from S800 show moderate variances regardless of the training model, but evaluations against LINNAEUS exhibit substantial increases — from -8.71 pp to 4.99 pp vs. from -0.72 pp to 12.79 pp. The development setup of training on S800 and evaluating on LINNAEUS outperforms the test setup by more than 30 %. LINNAEUS alone performs nearly 12 % better in development than in testing, while the original S800 shows only a 2.25 % difference, underscoring the limited internal coherence of LINNAEUS. This prompts a reconsideration of the

train-dev-test split in the LINNAEUS dataset.

Interestingly, when using LINNAEUS for training and S800 for evaluation, the relative performance shifts from positive to negative, indicating that the model performs better in the test set than in the development set, even though it was optimized for development. This trend also occurs with the filtered version of LINNAEUS but is unique to this setup.

	Training Data				
	Development dataset				
Evaluation Data	**S800**	**LIN.**	**LIN. FILT.**	**S800 and LIN.**	**S800 and LIN.FILT.**
S800	95.97 % +21.48 pp	79.85 % +9.7 pp	82.36 % +10.54 pp	95.85 % +20.69 pp	95.49 % +19.71 pp
LINNAEUS	79.14 % +3.91 pp	97.12 % 0 pp		96.42 % +1.00 pp	
LIN. FILT.	92.62 % +9.35 pp		96.34 % 0 pp		95.84 % +1.02 pp
	Test dataset				
S800	89.49 % +16.64 pp	81.56 % +15.76 pp	83.13 % +12.44 pp	91.49 % +17.71 pp	91.18 % +18.53 pp
LINNAEUS	60.21 % -9.76 pp	86.78 % 0 pp		86.56 % -1.72 pp	
LIN. FILT.	77.86 % -0.57 pp		80.51 % 0 pp		80.19 % +1.33 pp

Table 6.1: F1-scores and improvements for revised versus original S800 development and test datasets. In each cell, the top number shows the revised dataset's final F1-score, and the bottom number shows the percentage point difference from the original, with positive values indicating improvement.

	Training Data				
	Development vs. Test dataset				
Evaluation Data	**S800**	**LIN.**	**LIN. FILT.**	**S800 and LIN.**	**S800 and LIN.FILT.**
S800					
original	2.25 %	6.61 %	1.60 %	1.87 %	4.31 %
revisioned	7.24 %	-2.10 %	-0.93 %	4.77 %	4.73 %
difference	4.99 pp	-8.71 pp	-2.52 pp	2.90 pp	0.42 pp
LINNAEUS					
original	23.39 %	11.92 % (1)		8.09 %	
revisioned	31.44 %	11.92 % (1)		11.39 %	
difference	8.05 pp			3.30 pp	
LIN. FILT.					
original	6.71 %		19.66 % (1)		20.24 %
revisioned	18.96 %		19.66 % (1)		19.52 %
difference	12.79 pp				-0.72 pp

Table 6.2: F1-score summarization between the S800-orig test and devel datasets in percentage points. (1) These are in-corpus runs of LINNAEUS and thus S800 does not affect the results.

Impact of training size, category and other considerations

Overall, it is evident that the revision process has enhanced the S800 dataset's ability to identify species. The benefit of more precise annotation guidelines is particularly noticeable in instances where strains — mistakenly included in the test set as species — were not recognized by the updated model. However, this exclusion, although correct, incorrectly resulted in a decreased F-score.

One of the aspect of this study was to find out whether the revisioning has made **an improvement in in-corpus, cross-corpus and combined cross-corpus experiments. All of them show the same pattern: when evaluated against S800 data sets, the revisioned data set offers better results.** However, when employing the LINNAEUS dataset as an evaluation metric, the tests have not demonstrated any improvement. More specifically, the LINNAEUS test set significantly underperforms in comparison to the development set. This underperformance raises questions about the internal consistency of the LINNAEUS dataset.

In the fourth research question we were considering more extensive evaluation strategies than just the F1-score. We found out that a **notable portion of errors within the LINNAEUS dataset can be traced back to a single document, highlighting the limitations of using the average F-score as the sole performance metric.** This was also true to some extent with the S800, but obviously to a lesser effect since the S800 has a multiple of the number of documents compared to the LINNAEUS. **We attempted to find methods to identify these challenging documents. This was done by calculating F-score of each document and taking account also species proportions of that single document. The metric clearly showed documents, which performed poorly by the model.**

We were also interested in finding the effect of training size and categories of species in regards of the model performance — questions 2 and 3. Our findings reveal that **simply increasing the size of the training set does not directly translate to better performance. Notable improvements were observed with a minimal portion of the training data**, emphasizing the need to focus on addressing the more difficult cases instead of just expanding the training dataset. The achieved improvement varied a lot **within categories where protistology showing the most improvement, whereas virology remained stagnant**. Fu-

ture research should aim to delve into these discrepancies and develop more sophisticated metrics for evaluating model performance, which could provide deeper insights into the specific areas where the model struggles.

References

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.", *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] M. Minsky and S. Papert, "An introduction to computational geometry", *Cambridge tiass., HIT*, 1969.

[3] R. A. Fisher, "The use of multiple measurements in taxonomic problems", *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781*, 2013.

[5] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning", in *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010, pp. 384–394.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations", *arXiv preprint arXiv:1802.05365*, 2018.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training*, 2018.

[12] W. L. Taylor, ""Cloze procedure": A new tool for measuring readability", *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need", in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] E. F. Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition", *arXiv preprint cs/0306050*, 2003.

[15] J. Luoma and S. Pyysalo, "Exploring Cross-sentence Contexts for Named Entity Recognition with BERT", *arXiv preprint arXiv:2006.01563*, 2020.

[16] B. C. Gerner M. Nenadic G., "LINNAEUS: A species name identification system for biomedical literature", *BMC Bioinformatics 11, 85 (2010)*, 2010.

[17] D. Koning, I. N. Sarkar, and T. Moritz, "TaxonGrab: Extracting taxonomic names from text", *Biodiversity Informatics*, vol. 2, 2005.

[18] G. Sautter, K. Böhm, and D. Agosti, "A combining approach to find all taxon names (FAT)", *Biodiversity informatics*, vol. 3, 2006.

[19] E. Pafilis, S. P. Frankild, L. Fanini, S. Faulwetter, C. Pavloudi, A. Vasileiadou, C. Arvanitidis, and L. J. Jensen, "The SPECIES and ORGANISMS resources for fast and accurate identification of taxonomic names in text", *PloS one*, vol. 8, no. 6, e65390, 2013.

[20] N. Okazaki and S. Ananiadou, "Building an abbreviation dictionary using a term recognition approach", *Bioinformatics*, vol. 22, no. 24, pp. 3089–3095, 2006.

[21] S. Pyysalo, *Documentation for S800 corpus revision*, `https://spyysalo.github.io/s800-revision-docs/#comparison-of-annotations-between-s800-and-linneaus`.

Appendix A Appendix

A.1 Chapter 3 - Data

A.1.1 Documentation for S800 corpus revision

This is a plain copy from [21] extracted on 2021-12-11.

Documentation for S800 corpus revision

Revision steps

1. Decouple recognition from normalization, address overall consistency The original S800 corpus data only annotated mentions that could be normalized to a particular (2013?) version of the NCBI Taxonomy. From the perspective of recognizing mentions of species names (regardless of normalization), this caused the annotation to appear incomplete in places. In the first revision step, annotation was added for scientific and common names of species regardless of whether they could be normalized, a revision pass addressing the overall consistency of annotation was performed, and annotated names of genera, families and other levels of taxonomy above species were marked Out-of-scope.

Relevant commits: 59fe5be, 0165f52

2. Revise annotated spans for boundary consistency The original evaluation using the S800 corpus data applied relaxed boundary matching criteria:

[...] we used flexible boundary matching of species names, meaning that taggers would receive a true positive if it produced a tag that overlapped with an annotated substring and had the correct assigned taxonomic identifier.

Because of this matching criterion, the exact boundaries of annotated species mentions had little impact on the evaluation, and the annotation effort did not establish precise guidelines detailing which tokens should be included in annotated spans. Consequently, from the perspective of exact matching – the criterion used e.g. by the popular conlleval evaluation script and most recent machine learning ex-

periments on the corpus – the boundaries of annotated mentions were inconsistently annotated in many places. To address this issue, we created detailed guidelines on how to determine annotation boundaries and made a revision pass addressing related issues in the data. This revision also included a focused review of the annotation of virus mentions, which had comparatively frequent annotation boundary issues.

Relevant commits: 112f0b2, 7012e42

3. Separate strain mentions from species mentions The original S800 corpus annotation only involves a single annotated mention type (species) that is used to annotate mentions of species names as well as mentions of more specific names such as those of strains. In the revision, we introduced a separate Strain type and revised all strain name mention to use this type, also revising the spans of species annotation to exclude strain names when the two occurred together in text.

Relevant commits: 5212baf, 489fdf1

4. Introduce annotation for upper taxonomic ranks The original S800 corpus annotation included partial annotation for mentions of names at taxonomic ranks above species, in particular in a number of cases where these names were used in an imprecise way to refer to species (e.g. Drosophila for Drosophila melanogaster). As the initial revision step addressing annotation consistency had marked these as Out-of-scope, the coverage of the revised annotation was in some aspects reduced from that of the original corpus. In this revision step, annotation for mentions of names at taxonomic ranks above species was reintroduced in a systematic way, adding e.g. Genus as a distinct annotated type.

Relevant commits: 20ccfb7

Original Curator Guidelines for S800 corpus The annotation guidelines as described in the original publication of the S800 corpus

The guidelines to curators were to annotate all substrings, which can meaningfully be identified as referring to a taxon. While the main focus was on annotating

species mentions, strings referring to any taxonomic level, (e.g.kingdoms, orders, genera, strains) were also considered. These data were very likely never released.

The main guidelines were:

- All document substrings must be evaluated and all mentions including repetitions should be listed in the order of appearance in the text.
- The annotated name types among others should include: Linnaean binomials, common names, strain names, author defined acronyms.
- For each annotated string, curators must record the name as it appeared in text and report the corresponding NCBITaxonomy database identifier.
- Special cases of adjectives being used to indicate a taxon, misspellings, typographic or other errors and enumerations were indicated as such.
- Taxonomic mentions that did not correspond to an existing NCBI Taxonomy database entry were also indicated.

Inconsistencies detected that initially seem to contrast the above guidelines

- Some fairly obvious species names lack annotation for unidentified reasons. For examples, see mentions of Escherichia coli in 20971900 and 21324422, Pseudomonas aeruginosa in 21075931, and Flavobacterium sinopsychrotolerans in 20118284.
- Inconsistencies in tagging of strains (e.g. strains are not tagged in 20118285
- genera etc. tended not to have any annotation, e.g. 4/7 first documents (numeric order):
  - the genus Serinicoccus (x3) in 20118285

  - the genus Streptococcus (x2) in 20139281
  - the Bulleribasidium clade (x3) in 20139282
  - the genus Scardovia in 20139283

- There appears to be some inconsistency around the tagging of upper-level taxonomical terms such as virus and viral, which are tagged in 20980513 but not in over 100 other cases.

- The adjectival form pneumococcal is annotated in 7/12 cases in S800 although the general rule appears to be that adjectival forms are not annotated (e.g. murine, bovine, cyanobacterial) in contrast to guidelines.

- There is also an apparently inconsistent treatment of common species names, with e.g. mouse, swine, trout and rat partially annotated, but e.g. goat, horse, crab, and rats never receiving annotation in S800. (rat vs. rats being a particularly clear case)

A complete list of inconsistencies can be viewed in this table if it is filtered down to terms annonated in the S800 corpus. See also Comparison of annotations between S800 and LINNEAUS.

Other inconsistencies in the annotation These are mainly annotation boundary issues. Correct annotation boundaries was not a requirement in the original corpus annotation guidelines and was not taken into account when reporting F-scores on the datasets (*"For the mention-level statistics we used flexible boundary matching of species names, meaning that taggers would receive a true positive if it produced a tag that overlapped with an annotated substring and had the correct assigned taxonomic identifier. For example, if the string "E. coli K12" is annotated in S800 and the tagger matches only the string "E. coli", it will be counted as a true positive (provided the taxonomic identifier is also correct)."*).

- The expressions sp. nov. and gen. nov., sp. nov. following a species name are included in annotated spans in some but not all cases. For example, these are part of the annotated spans at the start of documents 20118285, 20139281, and 20154326, but not 19667393, 20139283 and 20139284
- The string (T) appearing at the end of strain names (superscript T in the original text) is included in annotated spans in some but not all cases. For examples, compare e.g. 19667393 and 20118285.
- When the name of the person who first named a species is mentioned in parenthesis after the species name (e.g. Diabrotica virgifera virgifera (LeConte)), this is annotated in some but not all cases. Also, the same question arises for the non-parenthesized forms such as Pseudacteon tricuspis Borgmeier.
- tomato vs. tomato plant(s)
- HCV vs. anti-HCV
- niger vs. A. niger strain
- galaxias vs. native galaxias
- Salmonella enterica vs. Salmonella enterica serovar Typhimurium
- (this example is extra) Influenza vs. Influenza vaccine

Additional guidelines for reannotation of the S800 corpus

- The first resource that is trusted to resolve issues is NCBI Taxonomy. If there is still not enough information there then the Catalogue of Life is advised

Span consistency guidelines

- Do not include the expressions sp. nov. and gen. nov., sp. nov. in the species name, since these are supposedly used only the first time a genus and/or a

species/subspecies is described to denote that it's new, so they are not part of the scientific name and shouldn't be found anywhere else other than the first paper describing them.

- An annotated span should not end with sp. or spp.
- Superscript T to denote type strain should not be included in species' names
- The person's name should not be included in the species name, especially when it is in parentheses. The non-parenthesized form is a bit more complex (at least in the example above Pseudacteon tricuspis Borgmeier is a valid name shown as a synonym for Pseudacteon tricuspis in NCBI taxonomy). For annotation consistency the suggestion is to drop these names in all appearances. (The confusion with subspecies can be avoided because of the capital letter at the start of the second word, e.g. Ursus arctos arctos would be easy to distinguish from Ursus arctos Linneaus and then drop the name for the latter.)
- Do not include common head nouns such as "plants" in annotation span
- Do not include adjectival premodifiers such as "native" in annotation span
- Model words like SCID mouse should be excluded from annotations
- "species complex" should not be part of a species name, e.g. from 20682355

  The splicing activity of the PRP8 intein from the (species)B. dermatitidis, (species)E. parva and (species)P. brasiliensis species complex was demonstrated in a non-native protein context in (species)Escherichia coli.
- f. sp. (forma specialis) should be included in the annotated mention (e.g. Blumeria graminis f. sp. tritici)

- Do not include nouns identifying levels of taxonomy in annotation spans. For example, the words strain, serotype, serovar, and serogroup should be excluded from the spans of annotated Strain mentions. e.g from 20154326

  Strain (GSW-R14 (T) exhibited 97.6 % 16S rRNA gene sequence similarity ...

- Annotate antibodies e.g anti-HCV with species annotation for the organism (HCV) and Note: "anti-" prefix

general annotation guidelines

- Preprocessing errors (e.g. amp;) should be fixed
- Clade mentions will receive Clade normalizations and will be assigned type according to nearest non-Clade ancestor
- Similarly, no rank mentions will receive no rank normalizations and will be assigned type according to nearest ranked ancestor
- Mentions that are not monophyletic (e.g. fish) should be annotated as OOS with Note: not monophyletic
- Adjectival forms like murine (taxid:10090), bovine (taxid:9913), pneumococcal (taxid:1313) that map to a specific species should be annotated as such
- Adjectival forms of Phyla (e.g. cyanobacterial: taxid:1117) can only be annotated as OOS or not be annotated at all
- Adjectival forms of Kingdoms (e.g. viral, bacterial) can only be annotated as OOS or not be annotated at all
- Introduce a flag-attribute for cannot be normalized for cases that are not full names (but only understandable as references in context) e.g. strips of types O, A and Asia 1

- Non-name mentions (e.g. woman) and species clues (e.g. patients, children, men, women) should not be annotated. This includes the non-name mention man which should not be annotated as a synonym for Homo sapiens (taxid: 9606)

- The role in which common species names are mentioned should not be taken into account and all species names mentions should be annotated (e.g. rice mentioned as food or tobacco as cigarettes should still be annotated).

- Genus or higher level mentions (e.g. Arabidopsis, yeast) should only be annotated as the real taxinomic level (i.e. genus, phylum) and not as synonyms of species names. (e.g. Arabidopsis should be annotated as Genus and assigned the genus taxid:3701)

  The second face of a known player: *(Genus)Arabidops* *is silencing suppressor AtXRN4 acts organ-specifically*

- Former Species annotations that belong to the following taxinomic ranks: Class, Order, Family and Genus have been annotated as the latter in the corpus. Ranks higher than Class (e.g. Phylum, Kingdom) should receive an Out-of-scope annotation (NCBI Taxonomy Ranking adopted from Schoch, et. al, 2020

- For annotations above Species only the "coarse" ranks should be considered, thus mapping mentions at fine-grained levels to their coarse equivalents, e.g. Subgenus -> Genus, Subfamily -> Family etc. Some examples are given below:

  - Subfamily: Plusiinae –> Family
  - Superfamily: butterflies –> Family
  - Infraorder: snakes –> Order
  - Infraorder: planthoppers –> Order

  - Suborder: true bugs –> Order
  - Suborder: Heteroptera –> Order
  - Infraorder: dragonflies –> Order

- Tribe and Subtribe are also normalized to Family, while Cohort is normalized to Class

- For Subspecies mentions: when a subspecies name immediately follows a species name the entire mention is simply annotated as one slightly longer Species mention, e.g. Phocoenoides dalli dalli annotated as Species + taxid: 9745 (Rank: Subspecies).

- Biotypes should be treated the same way as Subspecies

- common name (scientific name)" mentions should be annotated as two mentions e.g from 21054435:

  We studied seasonal dynamics in deltaÎ3C of CO2 efflux (deltaÎ3C(E)) from non-leafy branches, upper and lower trunks
  and coarse roots of adult trees, comparing deciduous
  (species)Fagus sylvatica
  ((species)European beech) with evergreen
  (species)Picea abies ((species)Norway spruce).

- Forms identified by place names, like ecotype, are not annotated.

  For investigating cadmium uptake, we incubated protoplasts obtained from leaves of (species)Thlaspi caerulescens (Ganges ecotype) with a Cd-specific fluorescent dye.

- Cultivars should be annotated as OOS.

  The physiological traits underlying the apparent drought resistance of '(Out-of-scope) Tomatiga de Ramellet' (TR) cultivars.

- Rootstocks should be annotated as OOS e.g. in 20837155
- Non-taxonomic groupings such as Gram-positive/negative bacteria, marine bacteria or enteric bacteria should not be annotated. e.g.

  The redox-sensitive transcription factor SoxR in enteric bacteria senses and regulates the cellular response to superoxide and nitric oxide.

- The last rule includes cases like the following from 21037015

  (Species)Oscillochloris trichoides is a mesophilic, filamentous, photoautotrophic, nonsulfur, diazotrophic bacterium which is capable of carbon dioxide fixation via the reductive pentose phosphate cycle and possesses no assimilative sulfate reduction.

- tree and bush are non-taxonomic mentions and thus not annotated or annotated as OOS + Note: non-taxonomic
- Species names in noun phrase premodifier positions (e.g. Arabidopsis EDR1, Aspergillus nidulans cells) also in cases where they appear as part of the name of an entity of a non-organism type (e.g. human epidermal growth factor receptor 2 (HER2)) are annotated.
- Species names are annotated when they are part of hyphenated compound words (e.g. human-infecting) but NOT when they appear as a substring in a word not separated by a boundary such as a hyphen (e.g. nonhuman)
- Abbreviations are marked if the abbreviation stands for an organism mention in scope of the annotation, but not if the full form merely includes an organism mention e.g. in modifier position. For example, the H in HER2 is not annotated despite it standing for human.
- Standalone alga (algae, microalgae, macroalgae): OOS + Note: non-taxonomic

e.g. Algae is an informal term for a large and diverse group of photosynthetic eukaryotic organisms.

- protist (any eukaryotic organism that is not an animal, plant, or fungus) is a non-taxonomical expression and will be annotated as OOS + Note: non-taxonomic e.g.
- protozoa will also be annotated as OOS + Note: non-taxonomic
- methanotroph is a non-taxonomical expression and will be annotated as OOS + no taxid e.g.
- methanogen(s) over 50 Archaea species: annotated as OOS with Note: not monophyletic
- prokaryotes includes Bacteria and Archaea in the current three-domain system, so this will be annotated as OOS + no taxid, despite the fact that eukaryotes will be annotated as OOS + taxid:2759 e.g. and e.g.
- heterokonts and alveolates are clades of microorganisms and will be annotated as OOS + taxid:33634 and taxid:33630 respectively e.g. and e.g.
- cyanobacteria, eubacteria and the like should be annotated as OOS unless it's clear from context that the reference is definitely to the genus Cyanobacterium or Eubacterium respectively.
- Young animals (e.g. chicks, calfs etc) should NOT receive an annotation or should receive an OOS annotation
- Non-taxonomic groupings of organisms by their behaviour (e.g. herbivores, predators, parasites) are OOS in the annotation actinorhiza(l), mycorrhiza(l), ectomycorrhiza: OOS + Note: non-taxonomic
- species complex and clonal complex rank: OOS

Strains

- Strain aliases such as CC-12301(T) (=DSM 45298(T) =CCM 7727(T)) should be annotated in all instances as type Strain.
- name strain mentions should be annotated as two mentions of Species+Strain, e.g. from 20154326

  Strain (species)GSW-R14 (T) exhibited 97.6 % 16S rRNA gene sequence similarity to (species)F. gelidilacus (strain)LMG 21477 (T) and similarities of 91.2-95.2 % to other members of the genus Flavobacterium
- mentions of the form [Genus] sp. [Strain], should have a separate Genus and Strain annotation e.g.
- descriptive references to Strains using gene names are not annotated as organisms e.g. 21097612

Viruses

- Viruses (or other taxonomic units) that have species level of entry as "unidentified" (e.g. "retrovirus" taxid:31931 ("unidentified retrovirus" equivalent: "retrovirus") or "adenovirus" taxid:10535 ("unidentified adenovirus" equivalent: "adenovirus")) should NOT be annotated in the species level, but should be annotated at the higher taxonomic rank that better describes them (e.g. family rank for Retroviridae, Adenoviridae etc)
  - "virus"/"viral" OOS+taxid:10239 "Viruses" superkingdom
  - "retrovirus" Family+taxid:11632 "Retroviridae" family
  - "influenza virus" Family+taxid:11308 "Orthomyxoviridae" family
  - "herpesvirus" Family+taxid:10292 "Herpesviridae" family
  - "adenovirus" Family+taxid:10508 "Adenoviridae" family

  - "baculovirus" Family+taxid:10442 "Baculoviridae" family
  - "reovirus" Family+taxid:10880 "Reoviridae" family
  - "norovirus" Genus+taxid:142786 "Norovirus" genus
  - "ebola virus" Genus+taxid:186536 "Ebolavirus" genus
  - "cytomegalovirus" Genus+taxid:10358 "Cytomegalovirus" genus

- dengue: dengue is synonym for dengue fever (disease), annotate as OOS + no taxid unless dengue virus is mentioned when it should be annotated as taxid:12637 (species)

- smallpox: smallpox is synonym for smallpox disease, annotate as OOS + no taxid unless smallpox virus is mentioned when it should be annotated as taxid:10255 (species)

- influenza: influenza is synonym for the flu (disease), annotate as OOS + no taxid unless influenza X virus is mentioned when it should be annotated as Species. EXCEPTION: standalone influenza may be marked when organism sense is clear from context (e.g. infulenza strains)

- human adenovirus (or similar cases): when a mention cannot be normalized in an "identified" virus species it should be annotated e.g. as Species+taxid:9606 (Homo sapiens) for human and Family+taxid:10508 (Adenoviridae) for $_a$*denovirus*

Yeasts

- Discontinuous entities should be annotated as such (e.g. http://ann.turkunlp.org:8088/index.xhtml/S800/20933017?focus=610 643)

- all text spans including "yeast" should have an Out-of-scope annotation if the taxonomy level is higher than Species:

  - standalone yeast: OOS+taxid:147537 ("true yeast" subphylum) (Note: an even higher level may be included)
  - black yeast: Order+taxid:34395 ("black yeast" order)
  - budding yeast: Order+taxid:4892 ("budding yeasts" order)
  - fission yeast: Family+taxid:4894 ("fission yeasts" family)
  - truffle: Genus + taxid:36048 (Tuber genus)

Amoebae

- All amoebae instances have been revised to resolve confusion of non-taxonomical expression amoebae (type of cell or unicellular organism which has the ability to alter its shape), of taxid:554915 (OOS: Clade: Amoebozoa), and taxid:55774 (Genus: Amoebae). Most of the cases were non-taxonomical expressions (OOS + no taxid)
- testate amoebae: very common combination of mentions, which means shelled amoebae, which explains the form of microorganism(s): OOS + no taxid
- Interesting article 21112814, where both non-taxonomical and Genus amoebae are mentioned (only one real "amoebae" Genus in the corpus)

Common names

- In general, when a species and a higher-level entry in the taxonomy (e.g. genus) share a common name or synonym, the species interpretation should be preferred when it is not clear from context which is intended.
- Common names like human, goat, horse, and rats should be always annotated.
- Common names that should not be annotated in the species level:
  - fire ant: Genus and taxid:13685 (Solenopsis); Note: red fire ant, little fire ant, black fire ant etc should be tagged as the corresponding species)

- ant(s): Family+taxid:36668 (Formicidae)
- insect(s): when standalone assign Class+taxid:50557 (Insecta)
- sunflower: Genus+taxid:4231 (Helianthus)
- galaxias : Genus+taxid:51242 (Galaxias)
- mite: Class+taxid:6933 (Acari subclass)
- trout: several species of fish, annotate as OOS + no taxid
- leafminer and leaf miner: insects that eat the tissue of plants, annotate as OOS + Note: non-taxonomic
- fishes: OOS (Clade-like concept, non-tetrapoda vertebrata)
- bug: OOS + Note: non-taxonomic
- field cricket: OOS + Note: non-taxonomic
- mirid bug: Family+taxid:30084 (Miridae)
- clownfish: Family+taxid:30863 (Pomacentridae)
- elephant: 3 species, not monophyletic (both Elephas and Loxodonta genera), annotate as OOS + no taxid
- crab: infraorder containing 850 species, so it should be annotated as Order + taxid:6752 (Brachyura)
- grass: Family+taxid:4479 (Poaceae)
- seabird(s): OOS with Note: non-taxonomic
- marsupial (animals carry the young in a pouch) is a mammalian clade, e.g. and will be annotated as OOS + taxid:9263
- coral(s): Hexacorallia + Octocorallia, but paraphyletic because sea anemones are also part of Hexacorallia: annotated as OOS with Note: not monophyletic

  - DNA viruses, RNA viruses map to no rank entries: annotated as Kingdom + normalization to 2080735 and 2559587, respectively
  - dsRNA mycoviruses: OOS with Note: non-taxonomic
  - cereal: OOS with Note: non-taxonomic
  - kittiwake: OOS and Note: non-taxonomic

- Common names that should be annotated in the species level (but could be annotated in a higher taxonomic level)
  - rat: synonym for Rattus norvegicus and Rattus. Should be annotated as Rattus norvegicus (taxid:10116), unless explicitly referring to a different taxonomic unit (e.g. cotton rat: Genus + taxid:42414 (Sigmodon))
  - fruit fly: synonym for Drosophila melanogaster and Drosophila genus and Tephritidae family. Should be annotated as Drosophila melanogaster (taxid:7227), unless explicitly referring to a different taxonomic unit
  - bee: synonym for Apis mellifera, and Apoidea superfamily. Should be annotated as Apis mellifera (taxid:7460), unless explicitly referring to a different taxonomic unit (e.g. bumble bee)
  - duck: synonym for Anas platyrhynchos, but can be a synonym for other Anatidae. Should be annotated as Anas platyrhynchos (taxid:8839), unless explicitly referring to a different taxonomic unit
  - midge: synonym for Chironomus thummi, but can refer to several species of flies. Should be annotated as Chironomus thummi (taxid:7154), unless explicitly referring to a different taxonomic unit

Very specific distinctions

- 4 mentions of "astomes" in this document 21398102 are OOS

- Astome ciliates in this document 21398102 are also OOS
- But some types of astome ciliates had been established as an order Astomatida
- FGSC should not be annotated as it refers to something that's out of scope, namely Fusarium graminearum complex 22004876
- Mentions of carnivores in 21323921 have been annotated as OOS, interpreting these to refer generally to meat-eating animals rather than the mammalian order Carnivora
- human and primates in a context of non-human primates are annotated as two mentions 21295520
- Dictyoptera in 19257902 is a clade including two orders Blattodea (cockroaches) and Mantodea (mantids). This has been annotated as OOS + taxid:6970 (Dictyoptera clade)
- termite in 19257902 might be a family Termitoidae (termites), even though that's no rank in NCBI taxonomy. Below Blattoidea superfamily, other sister nodes are family. I decided to annotate as Family + taxid:1912919 following the taxonomy presented in Cataloue of life
- 2435057 is discussing about retroviruses, but terminology there is quite old (published in 1987). ICTV (International Committee on Taxonomy of Viruses) was used to figure out how those viruses are called/classified in that period tracing its history.
- GII.4 in 20980508 has been annotated as species, following the general rule about Clade mentions
- arbuscular mycorrhizal fungi (AMF) e.g. in 20880038 OOS + Note: non-taxonomic

- tropical japonica rice (e.g. 20946420): following rule about no rank entries: normalization to 1736656 and type Species

Guidelines pending to be applied/decided upon

- mutant species annotation 21054438
- Enterohemorrhagic Escherichia coli in 21148732
- panther, panthers

Table A.1: Word occurrences of the original and revisioned datasets

Occurrences	**Revisioned**		**Cumulative sum**	**Original**		**Cumulative sum**
1	684	19.15 %	19.15 %	836	22.55 %	22.55 %
2	361	20.21 %	39.36 %	346	18.66 %	41.21 %
3	103	8.65 %	48.01 %	103	8.33 %	49.54 %
4	53	5.94 %	53.95 %	53	5.72 %	55.26 %
5	38	5.32 %	59.27 %	49	6.61 %	61.87 %
6	24	4.03 %	63.30 %	30	4.85 %	66.72 %
7	16	3.14 %	66.43 %	19	3.59 %	70.31 %
8	9	2.02 %	68.45 %	11	2.37 %	72.68 %
9	6	1.51 %	69.96 %	7	1.70 %	74.38 %
10	6	1.68 %	71.64 %	8	2.16 %	76.54 %
11	4	1.23 %	72.87 %	5	1.48 %	78.02 %
12	5	1.68 %	74.55 %	6	1.94 %	79.96 %
13	4	1.46 %	76.01 %	1	0.35 %	80.31 %
14	3	1.18 %	77.18 %	2	0.76 %	81.07 %

Continued on next page

Table A.1 – continued from previous page

Occur-rences	**Revisioned**		**Cumulative sum**	**Original**		**Cumulative sum**
15	4	1.68 %	78.86 %	6	2.43 %	83.50 %
16	1	0.45 %	79.31 %	4	1.73 %	85.22 %
17	8	3.81 %	83.12 %	1	0.46 %	85.68 %
18	1	0.50 %	83.62 %			
19	2	1.06 %	84.69 %	3	1.54 %	87.22 %
20	1	0.56 %	85.25 %	1	0.54 %	87.76 %
21				1	0.57 %	88.32 %
23	1	0.64 %	85.89 %	1	0.62 %	88.94 %
24	1	0.67 %	86.56 %	1	0.65 %	89.59 %
25				1	0.67 %	90.26 %
26	1	0.73 %	87.29 %			
27	1	0.76 %	88.05 %			
28	1	0.78 %	88.83 %	1	0.76 %	91.02 %
31				1	0.84 %	91.86 %
35				2	1.89 %	93.74 %
36	1	1.01 %	89.84 %			
38				1	1.02 %	94.77 %
44				1	1.19 %	95.95 %
47	2	2.63 %	92.47 %			
52	1	1.46 %	93.92 %			
54	1	1.51 %	95.44 %			
56				1	1.51 %	97.46 %
57	1	1.60 %	97.03 %			

Continued on next page

Table A.1 – continued from previous page

Occur-rences	**Revisioned**		**Cumulative sum**	**Original**		**Cumulative sum**
94				1	2.54 %	100.00 %
106	1	2.97 %	100.00 %			
Total	3572			3708		

Table A.2: Word occurrences of the LINNAEUS dataset

Occurrences	**Words count**	**in %**	**Cumulative % sum**
1	194	4.6%	4.6%
2	61	2.9%	7.4%
3	43	3.0%	10.5%
4	16	1.5%	12.0%
5	19	2.2%	14.2%
6	6	0.8%	15.0%
7	6	1.0%	16.0%
8	6	1.1%	17.1%
10	4	0.9%	18.1%
11	2	0.5%	18.6%
12	3	0.8%	19.5%
13	1	0.3%	19.8%
14	2	0.7%	20.4%
16	1	0.4%	20.8%
17	1	0.4%	21.2%
18	5	2.1%	23.3%

Continued on next page

TableA.2 – continued from previous page

Occurrences	**Words count**	**in %**	**Cumulative % sum**
19	3	1.3%	24.6%
22	1	0.5%	25.2%
23	3	1.6%	26.8%
27	2	1.3%	28.0%
29	2	1.4%	29.4%
31	4	2.9%	32.3%
32	2	1.5%	33.8%
37	1	0.9%	34.7%
38	1	0.9%	35.6%
40	1	0.9%	36.5%
42	1	1.0%	37.5%
44	1	1.0%	38.5%
45	1	1.1%	39.6%
47	1	1.1%	40.7%
51	1	1.2%	41.9%
53	1	1.2%	43.2%
56	2	2.6%	45.8%
62	1	1.5%	47.2%
65	3	4.6%	51.8%
66	2	3.1%	54.9%
71	1	1.7%	56.6%
80	1	1.9%	58.5%
81	1	1.9%	60.4%

Continued on next page

TableA.2 – continued from previous page

Occurrences	**Words count**	**in %**	**Cumulative % sum**
96	1	2.3%	62.6%
111	1	2.6%	65.2%
112	1	2.6%	67.9%
159	1	3.7%	71.6%
174	1	4.1%	75.7%
267	1	6.3%	82.0%
331	1	7.8%	89.7%
437	1	10.3%	100.0%
Total 2895			

A/turkey/OH/313053/04 (H3N2)
glassy-winged sharpshooter
A/Aichi/472/2009 (H1N1)pdm
Glassy-Winged Sharpshooter
Auricularia auricula-judae
Flavobacterium gelidilacus
Oscillochloris trichoides
A/Sakai/89/2009 (H1N1)pdm
Homalodisca vitripennis
Cotterillia bromelicola
Bifidobacterium bifidum
black-legged kittiwake
human immunodeficiency
Auritidibacter ignavus
grey-crowned babblers
Africanized honeybees
Delphacodes kuscheli
Thlaspi caerulescens
Methanoregula boonei
Pseudomonas syringae
G. intraradicesNH(4)
Venturia inaequalis
Mantamonas plastica
Elatobium abietinum
Simkania negevensis
Gryllus bimaculatus
H. cantabrigiensis
Grapholita molesta
Pilophorus typicus
Nocardia farcinica
Abeoforma whisleri
coxsackievirus A9
FtsZ-I374V strain
White-footed mice
monkeypox viruses
Chlorella strains
35S::prosys plant
long-tailed finch
blue-footed booby
marine bacterium
African elephant
worker honeybees

Amoebophrya spp.	Ebola viruses	D. algensis
sheeppox viruses	H1N1pdm virus	NUM 1615(T)
Penaeus vannamei	Asia 1/CHN/05	LMG 25435(T
human adenovirus	ATCC 13306(T)	hepatitis A
Loxigilla noctis	Pirum gemmata	influenza A
native galaxias	yellow morels	Parus major
Strains G203R-T	Fusarium spp	H9N2 virus
Dalbulus maidis	FGSC species	E. faecium
virus SIVmac239	NDV isolates	Asian rice
seasonal A/H1N1	JCM 16421(T)	serotype A
corn leafhopper	DSM 45359(T)	DSM 863(T)
HIV-1 CRF01_AE	Picea glauca	H5 viruses
H1N1 influenza	H5N1 viruses	fire ants
Asia 1/YNBS/58	black morels	West Nile
B. burgdorferi	Friend virus	L. crassa
Saccharum spp.	JCM 16107(T)	corn smut
cv Cho-Ko-Koku	DSM 21436(T)	WT plants
ATCC 19166(T)	CAIM 1449(T)	FGSC 9935
tomato plants	P. penardii	E viruses
V. inaequalis	hepatitis B	A. annua
CCUG 57943(T)	K-12 strain	

Words only in the revised version

Nocardioides pyridinolyticus	Trithigmostoma cucullulus	Tracheliastes polycolpus
Pseudoentodinium elephantis	Ofatulena duodecemstriata	Mingxiaea wuzhishanensis
Paraclausilocola constricta	Paraclausilocola elongata	Nocardioides dokdonensis
Cylindrocopturus adspersus	Coccinella septempunctata	Microterricola viridarii
	Nocardioides hankookensis	Methanobacterium veterum
	Leptinotarsa decemlineata	Riemerella anatipestifer

Flavobacterium frigoris	Crocidosema lantana	Varroa destructor
Phycicoccus dokdonensis	Kentrophoros flavus	D. tuberspinifera
Prasinoderma singularis	Mingxiaea foliicola	Blumeria graminis
Nocardioides aquiterrae	Mirodysteria decora	Spodoptera litura
Helicostomella subulata	Dermamoeba algensis	Labidus praedator
Acanthocephalus galaxii	Seychelles warblers	Trochilia minuta
Paratontonia gracillima	S. lavenduligriseus	Cryptantha flava
Phycicoccus bigeumensis	Gynaikothrips uzeli	Aspidisca cicada
Strombidium constrictum	Fusarium ussurianum	Dysteria legumen
Peridiniopsis penardii	Xylella fastidiosa	Nabis alternatus
Conotrachelus nenuphar	Ofatulena luminosa	B. bacteriovorus
Flavobacterium terrae	Platynota rostrana	Cotesia sesamiae
Hippodamia convergens	Fusarium asiaticum	Sorghum bicolor
Kentrophoros gracilis	Leifsonia ginsengi	D. hydrostatica
Opercularia coarctata	Paramecium aurelia	V. melanodiscus
Mingxiaea hainanensis	Holophrya discolor	S. canaliculata
Jiangella alkaliphila	Ephemerella ignita	rhesus macaques
Labedella gwakjiensis	N. pyridinolyticus	Thresher sharks
B. amyloliquefaciens	Xanthomonas oryzae	Bacillus cereus
Jiangella gansuensis	O. duodecemstriata	Jiangella alba
human papillomavirus	Fusarium gerlachii	B. rhizoxinica
Mingxiaea variabilis	G. intraradicesNH	P. fluorescens
Trichogramma minutum	Antirrhinum majus	T. grypoclunis
Rhizobium mongolense	Plagiocampa rouxi	Prunus persica
Asplanchna priodonta	Geocoris bullatus	Geleia simplex
Mingxiaea sanyaensis	Brassica oleracea	M. cerradensis
Dysteria proraefrons	Cytospora umbrina	S. montagnesi
Platynota helianthes	Acineria uncinata	Malus baccata
Streptococcus ratti	Amorbia concavana	T. polycolpus

L. incrustans	gypsy moths	rice frog
E. conspersus	guinea pigs	A. pernix
D. mulanensis	P. africana	D. biwae
R. mongolense	S. richteri	D. minor
friend virus	C. flavipes	Beta HPV
G. fujikuroi	S. wilberti	S. suis
Tiger sharks	field voles	
D. granifera	B. bifidum	
Sitka Spruce	B. pumilus	

A.2 Chapter 4 - Results

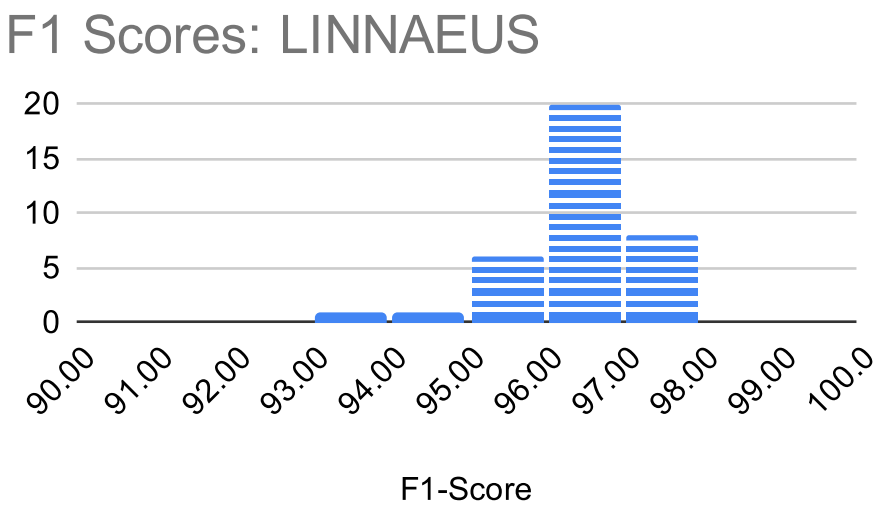

Figure A.1: LINNAEUS-filtered-corpus - devel.tsv: Histogram of the F1-scores.

Devel results

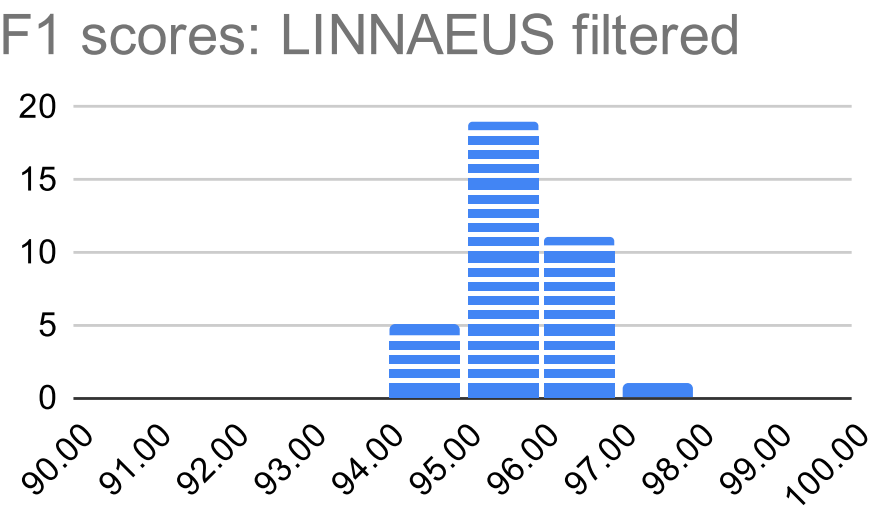

Figure A.2: LINNAEUS-corpus filtered/ - devel.tsv: Histogram of the F1-scores.

	precision	**recall**	**F-score**
Exact match	95.56% (516/540)	97.54% (516/529)	96.54%
Left boundary	95.93% (518/540)	97.92% (518/529)	96.91%
Right boundary	95.74% (517/540)	97.73% (517/529)	96.73%
Any overlap	96.11% (519/540)	98.11% (519/529)	97.10%

Table A.3: LINNAEUS-corpus filtered/ - devel.tsv: Evaluation metrics.

	precision	**recall**	**F-score**
LINNAEUS original			
Exact match	69.77% (494/708)	85.91% (494/575)	77.01%
Any overlap	71.05% (503/708)	87.48% (503/575)	78.41%
LINNAEUS filtered			
Exact match	91.85% (496/540)	79.74% (496/622)	85.37%
Any overlap	93.52% (505/540)	81.19% (505/622)	86.92%

Table A.4: Train: S800 original Devel: LINNAEUS : Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	70.20% (497/708)	93.77% (497/530)	80.29%
Any overlap	70.90% (502/708)	94.72% (502/530)	81.10%
LINNAEUS filtered			
Exact match	92.22% (498/540)	93.26% (498/534)	92.74%
Any overlap	93.70% (506/540)	94.76% (506/534)	94.23%

Table A.5: cross TRAIN S800 revisioned DEVEL LINNAEUS - devel.tsv: Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	74.74% (287/384)	69.16% (287/415)	71.84%
Any overlap	83.07% (319/384)	79.04% (328/415)	81.00%
LINNAEUS filtered			
Exact match	74.22% (285/384)	75.20% (285/379)	74.71%
Any overlap	82.55% (317/384)	84.70% (321/379)	83.61%

Table A.6: Train: LINNAEUS Devel: S800 original : Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	84.16% (287/341)	79.50% (287/361)	81.77%
Any overlap	87.98% (300/341)	83.66% (302/361)	85.76%
LINNAEUS filtered			
Exact match	86.22% (294/341)	84.00% (294/350)	85.09%
Any overlap	91.50% (312/341)	89.71% (314/350)	90.60%

Table A.7: cross TRAIN LINNAEUS DEVEL S800 1011 - devel.tsv: Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	77.60% (298/384)	75.06% (298/397)	76.31%
Any overlap	88.28% (339/384)	86.40% (343/397)	87.33%
LINNAEUS filtered			
Exact match	76.56% (294/384)	78.82% (294/373)	77.68%
Any overlap	87.24% (335/384)	91.69% (342/373)	89.41%

Table A.8: Train: LINNAEUS-and-S800 original Devel: S800 original : Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	95.31% (325/341)	97.89% (325/332)	96.58%
Any overlap	96.19% (328/341)	98.80% (328/332)	97.47%
LINNAEUS filtered			
Exact match	84.46% (288/341)	82.29% (288/350)	83.36%
Any overlap	91.50% (312/341)	89.43% (313/350)	90.45%

Table A.9: cross TRAIN LINNAEUS-and-S800 revisioned DEVEL S800 revisioned - devel.tsv: Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	96.19% (681/708)	95.92% (681/710)	96.05%
Any overlap	97.03% (687/708)	96.76% (687/710)	96.90%
LINNAEUS filtered			
Exact match	94.44% (510/540)	98.27% (510/519)	96.32%
Any overlap	95.00% (513/540)	98.84% (513/519)	96.88%

Table A.10: Train: LINNAEUS-and-S800 original Devel: LINNAEUS : Evaluation metrics.

	precision	recall	F-score
LINNAEUS original			
Exact match	96.05% (680/708)	97.00% (680/701)	96.52%
Any overlap	96.89% (686/708)	97.86% (686/701)	97.37%
LINNAEUS filtered			
Exact match	95.37% (515/540)	96.80% (515/532)	96.08%
Any overlap	96.11% (519/540)	97.56% (519/532)	96.83%

Table A.11: cross TRAIN LINNAEUS-and-S800 revisioned DEVEL LINNAEUS - devel.tsv: Evaluation metrics.

A.2.1 Fuzzy tags

Table A.12: Caption

	$S800_S$	$S800_O$	$S800_{Op}er_SplusO$	$S800_Oimp_05$
yeast	31	14	0.311	28.000
Arabidopsis	14	15	0.517	28.000
mice	22	10	0.312	20.000
rice	25	10	0.286	20.000
human	51	6	0.105	12.000
influenza virus	5	7	0.583	10.000
Drosophila	8	4	0.333	8.000
trout	6	4	0.400	8.000
wheat	6	4	0.400	8.000
HIV	20	4	0.167	8.000
A	4	144	0.973	8.000
O	3	14	0.824	6.000
sunflower	3	10	0.769	6.000
HCV	15	3	0.167	6.000
cotton	3	2	0.400	4.000
FV	2	3	0.600	4.000
fire ant	2	11	0.846	4.000
HBV	11	2	0.154	4.000
B. tabaci	11	2	0.154	4.000
yeasts	2	3	0.600	4.000
maize	14	2	0.125	4.000
Sinorhizobium meliloti	3	2	0.400	4.000

Continued on next page

Table A.12 – continued from previous page

	$S800_S$	$S800_O$	$S800_{Op}er_SplusO$	$S800_Oimp_05$
adenovirus	2	4	0.667	4.000
mouse	13	2	0.133	4.000
C. albicans	10	1	0.091	2.000
HDV	10	1	0.091	2.000
Escherichia coli	33	1	0.029	2.000
human adenovirus	1	1	0.500	2.000
VV	4	1	0.200	2.000
Bt	1	1	0.500	2.000
B. bassiana	3	1	0.250	2.000
native galaxias	1	1	0.500	2.000
Osspr1	1	1	0.500	2.000
T4	1	1	0.500	2.000
Mycobacterium tuberculosis	3	1	0.250	2.000
Cryptococcus neoformans	7	1	0.125	2.000
black	1	7	0.875	2.000
truffle	3	1	0.250	2.000
Solenopsis invicta Buren	1	1	0.500	2.000
class I	4	1	0.200	2.000
L. crassa	1	1	0.500	2.000
Yeast	1	1	0.500	2.000
serotype A	1	2	0.667	2.000
JC polyomaviruses	2	1	0.333	2.000
A. niger strain	1	1	0.500	2.000
V. inaequalis	1	1	0.500	2.000

Continued on next page

Table A.12 – continued from previous page

	$S800_S$	$S800_O$	$S800_{Op}er_SplusO$	$S800_Oimp_05$
marine bacterium	1	2	0.667	2.000
Flavobacterium gelidilacus LMG 21477(T)	1	1	0.500	2.000
JEC21	6	1	0.143	2.000
pigs	1	2	0.667	2.000
corn smut	1	1	0.500	2.000
DAstV	4	1	0.200	2.000
Pseudacteon tricuspis Borgmeier	1	1	0.500	2.000
Serinicoccus profundi sp. nov.	1	1	0.500	2.000
peach	3	1	0.250	2.000
H5N1	1	1	0.500	2.000
poultry	1	3	0.750	2.000
H1N1	1	2	0.667	2.000
fire ants	1	3	0.750	2.000
virus SIVmac239	1	1	0.500	2.000
Penaeus stylirostris	1	1	0.500	2.000
tomato	20	1	0.048	2.000
M. anisopliae	2	1	0.333	2.000
G. molesta	4	1	0.200	2.000
SIV	2	1	0.333	2.000
hepatitis C virus	1	1	0.500	2.000
Venturia inaequalis	1	1	0.500	2.000
S. meliloti	3	1	0.250	2.000
human adenoviruses	2	1	0.333	2.000

Continued on next page

Table A.12 – continued from previous page

	$S800_S$	$S800_O$	$S800_{Op}er_Splus O$	$S800_O imp_0 5$
Penaeus vannamei	1	1	0.500	2.000
Elatobium abietinum	1	1	0.500	2.000
C. neoformans	8	1	0.111	2.000
Friend virus	1	1	0.500	2.000
NDV isolates	1	1	0.500	2.000
tomato plants	1	5	0.833	2.000
U. maydis	5	1	0.167	2.000
anti-HCV	1	4	0.800	2.000
Rumex palustris	1	0	0.000	0.000

A.2.2 In-corpus experiment

S800

Table A.13: Some false positives with their frequencies

S800 test.tsv evaluation run

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
3 A virus	9 mice
2 WN	6 influenza A H1N1 (2009)
2 West Nile (	6 HIV
2 . venerealis	5 pneumococcal
2 tawny	5 horses
2 coxsackievirus	4 truffle
2 bean	4 6C
2 Australian cuttlefish	4 .

1 WSN	3 T . dvoinosi
1 WN) virus	3 rice
1 - Winged Sharpshooter	3 P . penardii
1 virus	3 influenza A
1 swine - origin influenza A virus	3)
1 Spruce	2 WNV
1 S -	2 WN
50 total	**275 total**
Some samples only in revisioned	**Some samples only in original**
3 A virus	9 mice
2 West Nile (	6 influenza A H1N1 (2009)
2 . venerealis	6 HIV
2 tawny	5 pneumococcal
2 coxsackievirus	5 horses
1 WSN	4 truffle
1 WN) virus	4 6C
1 - Winged Sharpshooter	4 .
1 virus	3 T . dvoinosi
1 Spruce	3 P . penardii
1 S -	3)
1 rice blast pathogen	2 WNV
1 porcine endogenous retrovirus	2 vole
1 Porcine	2 T . zuze
1 PERV	2 tuberculosis
37 in total	**259 in total**

Table A.14: Some false negatives with their frequencies

S800 test.tsv evaluation run.

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
10 influenza A	5 M2T2B15
6 alfalfa	5 influenza A H1N1
5 influenza A virus	5 FGSC
5 broccoli	5 6C (T)
4 wSinvictaA	4 YC6903
4 stable fly	4 stable fly
4 PERV - B	4 LPU83
4 influenza A viruses	4 Cotterillia
4 equine	3 wSinvictaA
3 tobacco	3 tobacco
3 tawny owls	3 H1N1
3 PERV - A	3 alfalfa
3 CVB3	2 Thiomonas perometabolis
2 swine	2 Thiomonas intermedia
113 total	**170 total**
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
10 influenza A	5 M2T2B15
5 broccoli	5 influenza A H1N1
4 PERV - B	5 FGSC
4 equine	5 6C (T)

3 tawny owls	4 YC6903
3 PERV - A	4 LPU83
2 swine	4 Cotterillia
2 rice	3 H1N1
2 Kaposi ' s sarcoma - associated herpesvirus	2 Thiomonas perometabolis
2 coxsackievirus B3	2 Thiomonas intermedia
1 West Nile (WN) virus	2 S . fibuligera
1 S - OIV	2 Salmonella enterica
1 porcine endogenous retrovirus (PERV) - B	2 Saccharomycopsis fibuligera
1 Porcine endogenous retroviruses	2 Nocardioides caricicola sp . nov .
70 in total	**140 in total**

LINNAEUS

Table A.15: linnaeus-corpus filtered/ - test.tsv: Some samples with their frequencies of false negatives and

False Positives - Exact match	
Some samples from Original	**Some samples from filtered**
20 Drosophila	20 Drosophila
16 Woodpecker	10 Vibrio cholerae
10 Vibrio cholerae	7 Photobacterium profundum
8 Culex	6 V . vulnificus
8 - billed	6 murine
6 V . vulnificus	6 Human
6 murine	6 Culex
6 Ivory - billed	4 sand fly
5 sand flies	4 sand flies

5 Photobacterium profundum	3 worms
5 Human	2 rat
3 Woodpeckers	2 HumanPSD
3 sand fly	2 human
3 mosquito	2 Ascaris
3 Brown Swiss cattle	1 worm
170 total	**100 total**
Some samples from Original	**Some samples from filtered**
16 Woodpecker	1 worm
8 - billed	1 Streptococcus Pneumoniae
6 Ivory - billed	1 S .
3 Woodpeckers	1 Escherichia coli
3 Brown Swiss cattle	1 Crithidia luciliae
3 Brown Swiss	1 bacteriophage
3 -	
2 rodent	
2 patient	
2 avian	
2 Aedes	
1 woodpeckers	
1 woodpecker	
1 Vibrionaceae	
1 turanicus	
76 in total	**6 in total**

Table A.16: linnaeus-corpus filtered/ - test.tsv: Some samples with their frequencies of false negatives and false positives.

False Negatives - Exact match	
Some samples from Original	**Some samples from filtered**
47 Pileated Woodpecker	47 Pileated Woodpecker
44 Ivory - billed Woodpecker	44 Ivory - billed Woodpecker
27 Pileated Woodpeckers	27 Pileated Woodpeckers
8 calf	24 calves
5 Ivory - billed Woodpeckers	14 calf
5 chick	11 chick
4 V . vulnificus CMCP6	5 Ivory - billed Woodpeckers
4 persons	4 V . vulnificus CMCP6
4 flies	4 Photobacterium profundum SS9
3 Photobacterium profundum 3TCK	3 Photobacterium profundum 3TCK
3 cattle	2 zebrafish
3 calves	2 V . vulnificus YJ016
2 V . vulnificus YJ016	2 Vibrio cholerae V52
2 Vibrio MED222	2 Vibrio cholerae V51
2 Vibrio cholerae V52	2 Vibrio cholerae RC385
204 total	**234 total**
Some samples from Original	**Some samples from filtered**
4 persons	2 Goat
4 flies	2 cows
3 cattle	2 chicken
2 fly	2 Calves

2 child	2 bacteriophage M13
2 boy	1 murine
1 Women	1 man
1 rabbits	1 Haemogogus
1 men	1 fruitfly
1 C . vishnui	1 Chinese hamster
1 Child	1 canine hookworm
	1 bovine
22 in total	**17 in total**

A.2.3 Cross-corpus experiment

Train: S800 Devel:LINNAEUS

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
10 Vibrio cholerae	19 Drosophila
7 Photobacterium profundum	14 Simmental
7 billed Woodpecker	9 Ascaris
6 V . vulnificus	6 V . vulnificus
5 murine	5 Ivory - billed
5 Human	4 sand flies
105 total	**155 total**
Some samples only in revisioned	**Some samples only in original**
10 Vibrio cholerae	19 Drosophila
7 billed Woodpecker	14 Simmental
5 murine	5 Ivory - billed
4 Swiss	4 sand flies
4 CMV	3 retroviral
4 - billed Woodpecker	2 T7
55 in total	**102 in total**

Table A.17: Train S800 Evaluate on LINNAEUS - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
151 patients	151 patients
74 women	74 women
65 men	65 men
46 Pileated Woodpecker	47 Pileated Woodpecker
46 patient	46 patient
36 persons	36 persons
770 total	**736 total**
Some samples only in revisioned	**Some samples only in original**
8 yeast	4 human
4 Photobacterium profundum SS9	2 V . cholerae
2 zebrafish	2 L . whitmani
2 Vibrio cholerae V52	1 Lutzomyia whitmani
2 Vibrio cholerae V51	1 Lutzomyia intermedia
2 Vibrio cholerae RC385	1 L . intermedia
31 in total	**12 in total**

Table A.18: Train S800 Evaluate on LINNAEUS - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: S800 Devel:LINNAEUS filtered

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
15 - billed Woodpecker	20 Drosophila
10 Vibrio cholerae	6 Photobacterium profundum
8 Brown Swiss	5 V . vulnificus
7 Photobacterium profundum	5 Rockingham
6 V . vulnificus	5 billed
5 murine	4 Simmental
113 total	**161 total**
Some samples only in revisioned	**Some samples only in original**
15 - billed Woodpecker	20 Drosophila
10 Vibrio cholerae	5 Rockingham
5 murine	5 billed
5 billed Woodpecker	3 SS9
4 Woodpecker	3 retroviral
2 mice	3 Malton
60 in total	**112 in total**

Table A.19: Train S800 Evaluate on LINNAEUS filtere - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
47 Pileated Woodpecker	47 Pileated Woodpecker
35 Ivory - billed Woodpecker	27 Pileated Woodpeckers
27 Pileated Woodpeckers	25 calves
25 calves	19 calf
20 calf	16 Ivory - billed Woodpecker
11 cows	14 bovine
263 total	**224 total**
Some samples only in revisioned	**Some samples only in original**
8 yeast	6 human
6 Drosophila	2 V . cholerae
6 chick	2 mice
3 fly	2 cow
2 zebrafish	1 SV40
2 Vibrio cholerae V52	1 murine
43 in total	**16 in total**

Table A.20: Train S800 Evaluate on LINNAEUS filtere - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS Devel: S800

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
10 patients	14 mice
5 mice	13 influenza A
5 dengue	10 patients
4 influenza	8 ant
3 Rhizobium sp	7 ants
3 poplar	6 influenza
137 total	**369 total**
Some samples only in revisioned	**Some samples only in original**
3 Rhizobium sp	8 ant
3 influenza virus	7 ants
3 fire ant	6 HIV
2 yeast	5 ticks
2 Ochromonas sp	5 horses
1 Yeast	4 E . coli
34 in total	**245 in total**

Table A.21: Train S800 Evaluate on LINNAEUS - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	8 YC6903
12 influenza A	6 M2 (T)
7 HAV	5 M2T2B15
6 KSHV	5 FGSC
6 alfalfa	5 6C (T)
5 tobacco	4 white - footed mice
157 total	**210 total**
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	8 YC6903
5 tobacco	6 M2 (T)
5 pneumococcal	5 M2T2B15
3 VSV	5 FGSC
3 tawny owls	5 6C (T)
3 equine	4 LPU83
58 in total	**148 in total**

Table A.22: Train S800 Evaluate on LINNAEUS - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train:LINNAEUS filtered Devel:S800

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
10 ant	14 mice
8 dengue	9 influenza A
7 ants	7 HIV
5 mice	5 influenza
4 A virus	5 horses
3 virus	4 rice
111 total	**272 total**
Some samples only in revisioned	**Some samples only in original**
10 ant	5 influenza
7 ants	5 horses
4 A virus	4 P . penardii
3 fire ant	4 Neurospora
3 bromelicola	3 T . zuze
2 influenza virus	3 T . suwako
46 in total	**186 in total**

Table A.23: Train LINNAEUS filtered Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
17 influenza A	8 YC6903
15 GII . 4	6 M2 (T)
6 tobacco	5 tobacco
6 alfalfa	5 M2T2B15
5 pneumococcal	5 FGSC
5 influenza A virus	5 6C (T)
5 HAV	4 white - footed mice
154 total	**199 total**
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	8 YC6903
5 pneumococcal	6 M2 (T)
5 HAV	5 M2T2B15
4 equine	5 FGSC
3 tawny owls	5 6C (T)
2 ORFV	4 LPU83
59 in total	**136 in total**

Table A.24: Train LINNAEUS filtered Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

A.2.4 Combi-cross experiment

Train: LINNAEUS and S800 Devel:S800

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
3 WN	10 mice
2 Spruce	6 influenza A H1N1 (2009)
2 PERV	6 HIV
2 Neurospora	5 horses
2 Beech	5 pneumococcal
2 bean	5 6C
2 Australian cuttlefish	3 T . zuze
47 total	**280 total**
Some samples only in revisioned	**Some samples only in original**
2 PERV	10 mice
2 Neurospora	6 influenza A H1N1 (2009)
1 West Nile (	6 HIV
1 weasel	5 pneumococcal
1 venerealis	5 horses
1 porcine endogenous	5 6C
1 Porcine	3 T . zuze
22 in total	**256 in total**

Table A.25: Train LINNAEUS filtered nd S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
9 influenza A	5 influenza A H1N1
6 alfalfa	5 FGSC
4 PERV - B	5 6C (T)
4 wSinvictaA	4 tobacco
4 equine	4 LPU83
3 swine	4 Cotterillia
87 total	**155 total**
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
9 influenza A	5 influenza A H1N1
4 PERV - B	5 FGSC
4 equine	5 6C (T)
3 PERV - A	4 LPU83
3 swine	4 tobacco
2 PERVs	4 Cotterillia
68 in total	**141 in total**

Table A.26: Train LINNAEUS filtered nd S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS filtered and S800 Devel:S800

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
2 . venerealis	9 mice
2 Spruce	6 influenza A H1N1 (2009)
2 Neurospora	6 HIV
2 influenza A	5 pneumococcal
2 Beech	5 horses
2 bean	4 truffle
57 total	**262 total**
Some samples only in revisioned	**Some samples only in original**
2 . venerealis	9 mice
2 Spruce	6 influenza A H1N1 (2009)
2 Beech	6 HIV
1 Yeast	5 pneumococcal
1 West Nile (	5 horses
1 tropical japonica	4 truffle
27 in total	**232 in total**

Table A.27: Train LINNAEUS filtered and S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
11 influenza A	5 tobacco
6 tobacco	5 M2T2B15
6 alfalfa	5 influenza A H1N1
4 wSinvictaA	5 FGSC
4 PERV - B	5 6C (T)
83 total	**180 total**
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
11 influenza A	5 M2T2B15
4 PERV - B	5 influenza A H1N1
3 PERV - A	5 FGSC
2 influenza A virus	5 6C (T)
2 equine	4 LPU83
56 in total	**159 in total**

Table A.28: Train LINNAEUS filtered and S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS and S800 Devel:LINNAEUS

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
15 GII . 4	6 M2 (T)
11 influenza A	5 tobacco
6 tobacco	5 M2T2B15
6 alfalfa	5 influenza A H1N1
4 wSinvictaA	5 FGSC
4 PERV - B	5 6C (T)
83 total	**180 total**
Some samples only in revisioned	**Some samples only in original**
15 GII . 4	6 M2 (T)
11 influenza A	5 M2T2B15
4 PERV - B	5 influenza A H1N1
3 PERV - A	5 FGSC
2 influenza A virus	5 6C (T)
2 equine	4 LPU83
56 in total	**159 in total**

Table A.29: Train LINNAEUS and S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
47 Pileated Woodpecker	45 Pileated Woodpecker
42 Ivory - billed Woodpecker	27 Pileated Woodpeckers
27 Pileated Woodpeckers	20 calves
21 calves	19 Ivory - billed Woodpecker
16 calf	18 calf
10 chick	10 chick
247 total	**208 total**
Some samples only in revisioned	**Some samples only in original**
3 Vibrio MED222	2 V . cholerae
2 zebrafish	2 bovine
2 women	1 man
2 V . vulnificus YJ016	1 Lutzomyia whitmani
2 Vibrio cholerae V52	1 Lutzomyia neivai
2 Vibrio cholerae V51	1 Lutzomyia intermedia
28 in total	**14 in total**

Table A.30: Train LINNAEUS and S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

Train: LINNAEUS filtered and S800 Devel:LINNAEUS filtered

False Positives - Exact match	
Some samples from revisioned	**Some samples from original**
10 Vibrio cholerae	19 Drosophila
10 Drosophila	7 Ivory
7 Photobacterium profundum	6 V . vulnificus
6 V . vulnificus	6 murine
5 murine	6 Human
5 Human	5 Vibrio cholerae
98 total	**119 total**
Some samples only in revisioned	**Some samples only in original**
2 Swiss	7 Ivory
2 quail	3 sand flies
2 M13	3 Ascaris
1 turanicus	2 Woodpeckers
1 New Zealand albino rabbits	2 sand fly
1 ne	2 L .
22 in total	**36 in total**

Table A.31: Train LINNAEUS and S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix

False Negatives - Exact match	
Some samples from revisioned	**Some samples from original**
47 Pileated Woodpecker	47 Pileated Woodpecker
43 Ivory - billed Woodpecker	39 Ivory - billed Woodpecker
27 Pileated Woodpeckers	27 Pileated Woodpeckers
24 calves	19 calf
19 calf	17 calves
11 chick	16 L . whitmani
240 total	**245 total**
Some samples only in revisioned	**Some samples only in original**
2 Vibrio cholerae V51	16 L . whitmani
2 Vibrio cholerae 0395	5 L . intermedia
1 Yeast	2 Goat
1 rabbits	1 SV40
1 nude mice	1 Sus scrofa
1 human	1 pig
9 in total	**29 in total**

Table A.32: Train LINNAEUS and S800 Evaluate on S800 - test.tsv: Some samples with their frequencies of false negatives and false positives. See more detailed listing in appendix