HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty	Koulutusohjelma – Utbildningsprogram – Degree Programme	
Faculty of Arts	Master's Programme in Linguistic Diversity and Digital Humanities	
Opintosuunta – Studieinriktning – Study Track		
Language Technology		
Tekijä – Författare – Author		
Rafael Leal		
Työn nimi – Arbetets titel – Title		
Unsupervised zero-shot classification of Finnish documents using pre-trained language models		
Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä– Sidoantal – Number of pages
Master's Thesis	10.11.2020	61

Tiivistelmä – Referat – Abstract

In modern Natural Language Processing, document categorisation tasks can achieve success rates of over 95% using fine-tuned neural network models. However, so-called "zero-shot" situations, where specific training data is not available, are researched much less frequently. The objective of this thesis is to investigate how pre-trained Finnish language models fare when classifying documents in a completely unsupervised way: by relying only on their general "knowledge of the world" obtained during training, without using any additional data.

Two datasets are created expressly for this study, since labelled and openly available datasets in Finnish are very uncommon: one is built using around 5k news articles from Yle, the Finnish Broacasting Company, and the other, 100 pieces of Finnish legislation obtained from the Semantic Finlex data service. Several language representation models are built, based on the vector space model, by combining modular elements: different kinds of textual representations for documents and category labels, different algorithms that transform these representations into vectors (TF-IDF, Annif, fastText, LASER, FinBERT, S-BERT), different similarity measures and post-processing techniques (such as SVD and ensemble models). This approach allows for a variety of models to be tested.

The combination of Annif for extracting keywords and fastText for producing word embeddings out of them achieves F1 scores of 0.64 on the Finlex dataset and 0.73-0.74 on the Yle datasets. Model ensembles are able to raise these figures by up to three percentage points. SVD can bring these numbers to 0.7 and 0.74-0.75 respectively, but these gains are not necessarily reproducible on unseen data.

These results are distant from the ones obtained from state-of-the-art supervised models, but this is a method that is flexible, can be quickly deployed and, most importantly, do not depend on labelled data, which can be slow and expensive to make. A reliable way to set the input parameter for SVD would be an important next step for the work done in this thesis.

Avainsanat – Nyckelord – Keywords
NLP, space vector model, zero-shot classification, Finnish language, pre-trained language models

Säilytyspaikka – Förvaringställe – Where deposited
E-thesis

Muita tietoja – Övriga uppgifter – Additional information

UNIVERSITY OF HELSINKI
FACULTY OF ARTS
DEPARTMENT OF DIGITAL HUMANITIES

Unsupervised zero-shot classification of Finnish documents using pre-trained language models

Master's Thesis
Language Technology

Rafael Leal

10.11.2020

Supervisor: Yves Scherer

Abstract

In modern Natural Language Processing, document categorisation tasks can achieve success rates of over 95% using fine-tuned neural network models. However, so-called "zero-shot" situations, where specific training data is not available, are researched much less frequently. The objective of this thesis is to investigate how pre-trained Finnish language models fare when classifying documents in a completely unsupervised way: by relying only on their general "knowledge of the world" obtained during training, without using any additional data.

Two datasets are created expressly for this study, since labelled and openly available datasets in Finnish are very uncommon: one is built using around 5k news articles from Yle, the Finnish Broacasting Company, and the other, 100 pieces of Finnish legislation obtained from the Semantic Finlex data service. Several language representation models are built, based on the vector space model, by combining modular elements: different kinds of textual representations for documents and category labels, different algorithms that transform these representations into vectors (TF-IDF, Annif, fastText, LASER, FinBERT, S-BERT), different similarity measures and post-processing techniques (such as SVD and ensemble models). This approach allows for a variety of models to be tested.

The combination of Annif for extracting keywords and fastText for producing word embeddings out of them achieves F1 scores of 0.64 on the Finlex dataset and 0.73-0.74 on the Yle datasets. Model ensembles are able to raise these figures by up to three percentage points. SVD can bring these numbers to 0.7 and 0.74-0.75 respectively, but these gains are not necessarily reproducible on unseen data.

These results are distant from the ones obtained from state-of-the-art supervised models, but this is a method that is flexible, can be quickly deployed and, most importantly, do not depend on labelled data, which can be slow and expensive to make. A reliable way to set the input parameter for SVD would be an important next step for the work done in this thesis.

Acknowledgements

This project was made possible by funding from the Finnish Ministry of Justice within the Anoppi project[1] in partnership with the Semantic Computing Research Group (SeCo)[2] at the University of Helsinki (HELDIG - Helsinki Centre for Digital Humanities) and Aalto University. I would like to offer my special thanks to Yves Scherer from the University of Helsinki for his invaluable comments; Aki Hietanen and Tiina Husso, from the Ministry of Justice, and Eero Hyvönen, Jouni Tuominen, and the rest of the SeCo crew for their support.

[1] Automatic anonymisation and content description of documents containing personal data. https://oikeusministerio.fi/en/project?tunnus=OM042:00/2018

[2] https://seco.cs.aalto.fi

Contents

Chapter 1

Introduction

Text classification can be considered one of the success stories in current Natural Language Processing: results vary depending on models and datasets, but many papers present accuracy rates above 95%, and in some cases close to 100%, especially in tasks such as News Categorisation and Topic Classification (cf. survey in Minaee et al. (2020, 27–29)). That is the case at least under ideal circumstances: a solid pre-trained deep learning-based language model with abundance of high-quality training data for fine-tuning.

These conditions might not always be present in real-life situations, however. This thesis aims at exploring feebler situations, in which only some of the elements above occur: most of the language models featured here are indeed pre-trained based on deep learning techniques, but appropriate data for fine-tuning is missing. This work researches how these language models fare in text categorisation tasks when there is no training data to rely upon. Extensively learning on already-existing texts hopefully imbues the models with enough "knowledge of the world" and allow them to statistically model language accurately. This thesis probes the semantic payload contained in the language models under study, and explores possible ways to enhance them.

This work sprang up from a project that intends to create an exploratory search engine for Finnish laws; this accounts for many of the decisions taken. The focus is thus on pre-trained Finnish language models and data, which limits the choices about materials and models. Since Natural Language Processing (NLP) as a global endeavour is dominated by English, far less resources are dedicated to other languages, even well-established European languages with

millions of native speakers. Moreover, as a language, Finnish stands recognisably apart from the so-called "Standard Average European", since it is largely agglutinative, has 15 nominal cases and traditionally no articles. According to Schwartz et al. (2020, 1), "[w]ithin the NLP literature, agglutinative languages like Finnish and Turkish are commonly held up as extreme examples of morphological complexity that challenge common modelling assumptions". On the other hand, Finnish is not polysynthetic, a trait common among American indigenous languages in which words are composed of many semantic morphemes: its concept of "word" might be not too far removed from the major Indo-European language for standard NLP models – which are built mainly with those in mind – to work. Indeed, an important advancement in recent NLP models is subword segmentation, which is able to better model the kind of bound morphemes so pervasive in agglutinative languages.

Another recent breakthrough in NLP is the development of transfer learning, which converts the knowledge obtained in one task to another. Mastering transfer learning opened the doors for the creation of large language models that can be tailored afterwards for specific tasks, which usually perform better than those created for specific purposes. It is in this context that the commonly called "zero-shot" classification takes place.

According to Wang et al. (2019), zero-shot learning aims to learn a classifier that is able to categorise instances belonging to an unknown class when given labelled training instances (i.e., belonging to "known" classes). This definition makes it clear that zero-shot classification is usually implemented within a transfer learning framework, from "seen" classes to "unseen" ones, using class descriptions, relations among classes and domain knowledge (Zhang, Lertvittayakumjorn, and Guo 2019). In this case, zero-shot can be considered a specific case of transfer learning, and one that is widely used in the field of computer vision. However, this is not the case for the present project, since there is an acute lack of open labelled data in Finnish – hence the term "unsupervised", which means that the algorithms cannot rely on guidance (for example from a gold standard) to correct its behaviours and outcomes.

The development of massive language models has also created the possibility to explore them in alternative ways: directly accessing their internal representations for the data they are modelling. This thesis makes use of these techniques on newer models, as well as older alternatives, to research the possibility of classifying texts without the need for training examples or human supervision, given a pool of possible category labels: do Finnish pre-trained language models

contain enough semantic information to allow for zero-shot classification without any training data? Is it possible to enhance the semantic payload in these models by using different algorithms or combinations of algorithms? How can the semantic similarity best be assessed?

The main mechanism used in this thesis is the creation of vector space models in which texts and class labels are represented. On the one hand, a common vector space simplifies the comparison among these disparate elements, and is very flexible when it comes to altering the set of class labels used. On the other hand, however, this means that the same space has to contain representations for different linguistic levels (words, sentences, paragraphs), so building it is far from an obvious process. A common vector space also facilitates multi-label categorisation, in which one document can belong to different categories. Although this is a very useful feature for the search engine mentioned above, this thesis concentrates on multi-class classification, in which the documents are classified into one of the multiple classes available.

Architecturally, this project consists of four independent modules that can be assembled in different combinations, which allows for a variety of test scenarios. Document extractors are responsible for representing the document textually: they may output the whole text, a part of it, or for example transform it into keywords. This representation is turned into a vector using one of the language representation models available. Category retrievers are also responsible for textual representation, in this case for the category labels: they can be yielded as-is, or semantically enhanced. These category representations are also transformed into vectors using the same language representation model. Finally, a classifier compares documents to categories using a similarity algorithm and predicts a category for each of the documents.

These predictions are tested in this thesis using two custom-made datasets: one smaller, more experimental, containing Finnish law sections, and a larger one (split up in two) containing news articles from Yle, the public Finnish Broadcasting Company.

Chapter 2 describes the theoretical background of this thesis and the algorithms used in it. Chapter 3 illustrates past research in the field of zero-shot classification. Chapter 4.2 discusses how the datasets are created and how these algorithms are used to build the different modular blocks describes above, while chapter 5 reports how well they fare when faced with the datasets and their ground truths. Finally, chapter 6 summarises the overall conclusions.

The code used in this project is available at https://version.aalto.fi/gitlab/seco/zeroshot-classifier.

Chapter 2

Background

This section describes the foundations on which this work is based: its main algorithms, methods and theoretical structures.

2.1 Vector Space Model

VSM can be considered the overarching theoretical framework for this thesis. This is a Natural Language Processing method in which linguistic elements (usually words or sentences) are mapped onto an Euclidean space, so that each feature corresponds to a dimension in that space. Linguistic elements are thus represented by a numeric vector corresponding to the values of its features. This leaves room for using mathematical measures such as Euclidean distance or cosine similarity to measure element similarity or finding correlations among terms. This method was developed for the field of Information Retrieval by Gerard Salton on the second half of the 20th century (Dubin 2004).

In the categorisation work carried out by Wang et al. (2019), vector spaces can be *engineered* (designed by humans), in which case the dimensions have an explicit meaning, or *learned*, i.e., obtained via machine learning, so that the meaning of the different dimension are implicit in the model used. The former are more amenable to transformations and incorporations, but harder to set up, while the opposite is true for the latter.

In its most basic implementation, a vector space model can be built with word counts: each word constitutes its own dimension, and the corresponding vector position can be a boolean value (to indicate presence) or a frequency count. This method does not take synonymity into

account: texts with similar content but dissimilar words end up with completely unrelated representations. Distributional semantics offer a solution to this problem by considering words as the sum of their contexts: since semantically equivalent words usually appear in similar sentences, a statistical model that takes context into account will tend to represent them in a roughly equivalent manner. However, in using this technique the amount of data is crucial: these semantic correspondences only come to the fore when a sufficient amount of examples is used.

The remaining challenge when dealing with a vector space model for semantic text classification is in how to embed different levels of linguistic elements (words, sentences, texts) into the same space. Their features are inherently different, and language models are usually geared towards one of these levels. In order to be able to categorise texts using class labels, an effective solution to this must be found, since the similarity measures all work on linear algebra principles, which require the same dimensionality.

2.2 Similarity measures

These measures indicate how closely related the elements are from one another. These are measures taken on a geometrical space: the assumption is that in the case of a vector space built with semantic features in mind, a high similarity value might also reveal semantic similarities – in the case of this thesis, how closely related the document representations are to each category representation.

2.2.1 Dot product

This measure is simply the sum of the pairwise product of the elements of two vectors. The length of the vectors must be the same:

$$A \cdot B = \sum_{i=1}^{n} A_i B_i \tag{2.1}$$

Geometrically, this measure is equivalent to projecting one vector onto the other and multiplying their lengths. It tests if the two vectors tend to point in the same direction: perpendicular vectors have a dot product of zero and vectors pointing in a different direction will have a negative dot product.

2.2.2 Cosine similarity

Cosine similarity is one of the most widely used distance measures when working with vector spaces: Manning, Raghavan, and Schütze (2008) call it the "standard way" of measuring similarity. It indicates how acute the angle between two n-dimensional vectors is: a result of 1 means that the vectors are overlapping (pointing in the exact same direction) and -1 indicates opposite directions (regarding the *origin* point), while a cosine of 0 indicates orthogonality (no relation between the vectors).

The formula for cosine similarity is the following:

$$cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2.2}$$

where the numerator $A \cdot B$ is the dot product between the two vectors and the denominator $\|A\|\|B\|$ is the product between their magnitudes. The magnitude of a vector, also called *Euclidean length*, is the sum of its squared components. Expanding equation 2.2:

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.3}$$

The denominator has the effect of *length-normalising* the vectors, dampening their differences. This is important for example in the case of a standard count matrix, where document length may unduly influence the results. This is not necessarily the case for deep-learning model vectors, where these differences might be important. For this reason, both are being tested in this thesis.

2.2.3 Pearson's correlation

Pearson's correlation is a measure commonly used in statistics to indicate if two variables vary together, i.e., it measures the strength and direction of the relationship between the elements of two variables. The results vary from -1 to 1: a number closer to zero indicates a weaker linear correlation, while the extremes mark a stronger one. The polarity indicates if the correlation is in the same direction or in opposition.

Mathematically, it is defined as the covariance of two variables divided by the product of their individual standard deviations (σ):

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{2.4}$$

While covariance is calculated element-wise as follows:

$$cov(X, Y) = \sum_{i=1}^{n} (X_i - \mu_X)(Y_i - \mu_Y) \tag{2.5}$$

μ_X and μ_Y are their respective means. Thus, Pearson's correlation is directly proportional to the position of each component of a vector relative to the same component in another vector. It can then be used as a measure of how alike these vectors are.

2.3 SVD

Singular Value Decomposition is a matrix decomposition method that transforms a matrix into the product of three matrices:

$$M_{m \times n} = U_{m \times m} \Sigma_{m \times n} (V^T)_{n \times n} \tag{2.6}$$

U contains the left-singular vectors of the original matrix, Σ is a diagonal matrix consisting of its singular values and V^T holds its right-singular vectors. The outer product of the left-most column of U with the uppermost row of V^T, multiplied by the first (top left) value of Σ, yields a matrix that has some of the information contained in M: it is the best approximation of the original matrix using only two vectors and a value. Adding this first matrix to another one, calculated by the outer product of the second elements (second column of U, second value in Σ, second row in V^T) adds more of the information: it becomes the best approximation to the original using that many elements (two left-singular vectors, two right-singular vectors, and two single values). The more of these "triplets" are added, the more the final matrix resembles the original M matrix. And since the single values contained in Σ are ordered from highest to lowest, the first "triplets" are the ones that have the most impact, while the last ones will be comparatively – or even in real terms – insignificant.

Consequently, M gets decomposed into multiple Rank 1 matrices by finding linear dependences (or approximations) in the existing data. By disregarding the last elements of U, Σ and V^T, it is possible to reconstruct most of the information found in the original matrix, but with the advantage of having highly correlated data. This feature is used in the *Truncated SVD* method to reduce the dimensionality of the original matrix by only considering the first k single vectors:

$$M_{m\times n} \approx U_{m\times k}\Sigma_{k\times k}(V^T)_{k\times n} \tag{2.7}$$

The result is thus an approximation of the original matrix, containing most of its variance, that is nevertheless more compact. This may improve the final representations and, as a consequence, the results of the categorisation. The size of this resulting matrix depends on the chosen value for k. There are different techniques for choosing k, for example plotting the percentage of variance it explains and spotting sudden breaks.

A common application of SVD in NLP is in Latent Semantic Analysis, where it is used to obtain "hidden" topics from a document-term (frequency) matrix: the U matrix represents how each document correlates with each topic and the V^T matrix reveals how the topics relate to the terms, while the Σ matrix contains the importance of each topic – i.e., how much of the variance in the data can be explained by this particular "topic" or singular vector.

2.4 Linked Data

Linked Data ontologies are structured data that systematically link concepts, properties and relationships in order to recreate human knowledge in computer-friendly format. It is usually formatted using the Resource Description Framework (RDF) data model, which guarantees inter-operability among the different data providers. In many cases, the data portals offer freely available Open Data and an endpoint that can be queried using the SPARQL query language.

In this thesis, Linked Data provides both the raw Finnish Legislation text data and a means to augment the category labels using semantic information. The former is explained in the 4.1 Data section in the next chapter. For the latter, the specific ontology used in this project, KOKO (Frosterus et al. 2013), is a collection of Finnish ontologies published by the National Library

of Finland and located at the Finto service[1]. It links concepts to one another in three different categories: *broader*, *narrower* and *related* concepts (as well as translations and resources in other languages). For example, when queried, the term "linguistics" appears linked to the following concepts:

broader concept humanities
narrower concepts applied linguistics, biolinguistics, cognitive linguistics, etc.
related concepts colloquial language, grammar, history of linguistics, language research, etc.

These relations are used in this thesis as a tool to semantically augment class labels.

2.5 Language Representation Models

Language representation models are computational models that provide statistical representations of a language. They are used in this thesis to transform textual representations of both documents and category labels into vectors on a vector space.

2.5.1 TF-IDF

TF-IDF is a simple algorithm widely used in Information Retrieval. It finds statistically relevant words from a collection of texts based on two measures: *term frequency*, which measures how relevant a term is to a text (usually a raw or scaled count of the word frequency in a text), and the *inverse document frequency*, which provides a measure of how rare a term is within the whole collection of texts, under the assumption that frequent words carry less information than rarer ones (Jurafsky and Martin 2019).

The calculations are performed with the *scikit-learn* tools in Python (Pedregosa et al. 2011). This implementation defines $\text{tf}(t)$ as the number of times the term t appears in the document and $\text{idf}(t)$ as the following:

$$\text{idf}(t) = \log\frac{1+n}{1+\text{df}(t)} + 1 \tag{2.8}$$

where N is the total number of documents and $\text{df}(t)$ is the total number of documents containing the term t. The additions serve as smoothing, i.e., to prevent mathematical errors that

[1] https://finto.fi/fi/

can occur with null values. The final equation is then:

$$\text{tfidf}(t, d) = \text{tf}(t, d) \times \text{idf}(t) \tag{2.9}$$

As a final step, TF-IDF vectors are normalised by their Euclidean norm, also known as the L2 norm, in order to make frequencies relative within each document:

$$\|x\|_2 = \sqrt{x_1^2 + \cdots + x_n^2} \tag{2.10}$$

2.5.2 fastText

fastText (Bojanowski et al. 2017) is a tool developed by the Facebook Research team that utilises *word2vec* word embeddings, adding to it subword information. This addition makes the embeddings more robust for unseen words.

2.5.2.1 word2Vec

The highly influential word2vec framework (Mikolov, Sutskever, et al. 2013) uses shallow neural networks (containing one hidden layer) to extract word vectors from a text. This model follows the distributional hypothesis (Harris 1954), using the textual contexts in which the words appear as the basis to calculate its representation. The model is trained using a sliding window that peeks at each word's in-reach neighbours as it slides through the text and updates the joint probabilities for the words in question.

Word2Vec comes in two different versions: *Continuous Bag-of-Words (CBOW)*, whose purpose is to find context words given an input word, and *skip-gram*, whose aim is the contrary, i.e., find a word given a context. For each step taken by the sliding window, words not in context must also be updated. However, instead of using all leftover words as negative examples, the word2vec framework uses either Hierarchical Softmax or Negative Sampling as a means to decrease the workload. The former takes into account only a specific path in a binary tree, while the latter samples a limited amount of randomly chosen negative examples out of the not-in-context words. This makes the training much faster while mostly retaining the quality of the representations (Mikolov, Sutskever, et al. 2013). Subsampling of frequent words is also used, so that frequent words above a certain threshold may be discarded during training. Once

again, this speeds up the training step while significantly improving the accuracy of the model (Mikolov, Sutskever, et al. 2013, 5).

One important innovation brought by this paper was capitalising on the hidden states present inside the neural networks as a source of information. While the network is trained for the specific task of finding contexts for words or vice-versa, it was observed that these internal representations carry in themselves semantic information that can be used for other purposes. They show regularities that allow for certain kinds of inferences and vector compositionality, especially regarding inflectional morphology and named entities (Finley, Farmer, and Pakhomov 2017).

2.5.2.2 fastText

Bojanowski et al. (2017) introduced an extension to the skip-gram model in order to improve on one main shortcoming of the word2vec framework: it ignores word morphology, assigning a different vector to each word form instead of grouping paradigms together. The fastText model, on the other hand, represents words as a bag of n-grams; each n-gram receives its own vector, and the words are represented by the sum of these n-gram vectors. This mechanism also provides the means to build representations for words that are not in the training data, by breaking them up into chunks that the model is able to recognise. The smallest chunks are the letters present in the source texts.

Special symbols are added to the beginning and end of the words, which makes possible to distinguish prefixes and suffixes from other letter sequences. Each word is then decomposed into n-grams. For example, the word `<woman>` becomes `<wo`, `wom`, `oma`, `man`, `an>`.

In the end, each word vector is calculated as the sum of its n-grams representations. Given a word w, its n-grams $G_w \subset \{1, ..., G\}$ and their representations (z_g):

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c \tag{2.11}$$

In the standard implementation, the vectors have a dimension of 300, which means that this many neurons are being used internally to represent each word.

The authors report superior results to both skipgram and CBOW in a variety of tasks and languages, including human similarity judgements (for Arabic, German, English, Spanish, French,

Romanian and Russian, except in a dataset containing common English words), word syntactic analogy tasks (in Czech, German, English and Italian) and word semantic analogy tasks (in Czech and English, but not Italian or German – the latter suffered a significant drop).

Naturally, this model carries over some of the shortcomings presents in word2vec. For example, there is no mechanism to effectively track the order of the words (although the sliding window provides an idea about neighbouring words, which is the basis of the algorithm). And beyond the sliding window, there is essentially no mechanism at all, so longer-distance relationships are ignored.

2.5.3 Annif

Annif (Suominen 2019) is an open-source tool developed by the National Library of Finland for automating subject indexing and classification. By default, it uses bibliographic metadata from works found in the Finna portal ('Finna.Fi' n.d.), which aggregates digital databases provided by different Finnish organisations – as of this writing, 11 archives, 60 museums and 73 libraries, including special, public and University libraries. This software works as a multi-label classifier, which means that each document may be assigned a number of different labels. This differs from the more traditional multi-class classification methods, which aims to classify by choosing only the best-fitting class out of a list.

Annif leverages the subject headings found in manually indexed documents. The training dataset consists of for example book titles and their corresponding metadata in YSO keywords: *Horace* is paired with *antiquity*, *literary research*, etc. After training, it can be used to propose subject headings for documents. The National Library of Finland also maintains a REST API which offers access to automatic subject indexing from pre-trained projects in Finnish, Swedish and English.

Annif is able to use a number of different algorithms: some of those, such as TF-IDF or Maui, use a *lexical approach*, matching terms representing a document to a vocabulary of terms. This type of algorithm can only find word forms that are present in the texts: they are not able to extrapolate to related concepts or overall interpretations. *Associative approaches*, for example Parabel or Bonsai, may be able to find indirect correlations. Often, a combination of these algorithms via *ensembles* or *fusion architectures* yield the best results (Toepfer & Seifert(2018), cited in Suominen (2019)), since individual algorithms may make mistakes that end up be-

ing overruled by the ensemble. A *decision function* merges the predictions from the different algorithms and returns a combined result.

Two of the projects are used in this thesis: one is an ensemble of Parabel, Maui and TF-IDF, and the other is made up by Parabel alone. This algorithm is explained in the next subsection.

Parabel is an extreme multi-label classification algorithm developed with advertising in mind: the developer's objective is primarily to predict queries that might lead to an advertisement click on a search engine. This explains the focus of the algorithm on extremely large datasets (the pool of non-mutually-exclusive labels from which the algorithm annotates the data can be composed of millions of items) and speed (both during training and prediction).

This algorithm follows the 1-vs-All approach, in which a separate linear classifier is learned for each label. However, it organises the data points into a hierarchy so that similar labels are grouped together at the leaves. The labels in a leaf act as negative examples for each other's classifier. Negative examples as thus restricted to the most confusing labels in the pool.

Parabel uses an ensemble of 3 label trees, all of which work in a similar way. In each internal node, two independent linear classifiers decide if the data point should be sent left, right or in both directions. A data point may therefore be sent to multiple leaves in each tree. All the relevant 1-vs-All classifiers are then activated, and the final prediction is the average of the probabilities gathered from the different label trees.

2.5.4 LASER

The Language-Agnostic SEntence Representations (LASER) toolkit (Artetxe and Schwenk 2019) was developed by the Facebook Research team to create massively multilingual sentence representations. The aim of this project is to create language-independent embeddings by using the same vectorial representations for all 93 languages it integrates. The list includes mostly European and Asian languages from different language families[2]. For the purposes of this thesis, it is important to note that both English and Finnish are included in the list, and that the purpose of this project is to create *sentence* embeddings, not *word* embeddings.

[2]The list of supported languages can be found at https://github.com/facebookresearch/LASER#supported-languages.

The LASER toolkit uses an encoder-decoder architecture with a single Bidirectional LSTM. These Long Short-Term Memory neural networks (Hochreiter and Schmidhuber 1997) are special types of Recurrent Neural Networks (RNNs), which means that they process the input sequentially, one element after another, using the previous state as well as the new element. LSTMs are designed to mitigate the so-called *exploding of gradients* that can occur in RNNs, especially long ones, and result in loss of information. This is mitigated by equipping each cell in the network with a cell state vector, which control the flow of information from one cell to another, actively deciding what to remember and what to forget.

The encoder is responsible for transforming the sentences into vectorial representations, while the corresponding decoder, which is not used for obtaining embeddings, is capable of doing the inverse job. The embeddings are built using the same encoder for all languages. The vocabulary, also shared by all languages, consists of 50k tokens of byte-pair encoding obtained from the concatenation of the whole training corpora: similarly to fastText, each word is represented by subword chunks available in the model's dictionary. The encoder does not take into account the language of the training sentences in order to create language-agnostic representations. This architecture is trying to represent the pure meaning of the sentences by disregarding their encoding – i.e., language (Schwenk 2019).

Each direction of the bidirectional encoder (left-to-right and right-to-left, as input sentences are processed) has a dimension of 512; the final representations concatenates both directions, which results in a dimension of 1024. A bidirectional architecture is a method used to take into account both preceding and following words in a sentence. The authors believe that fixed-length embeddings are more versatile and compatible (Artetxe and Schwenk 2019).

The training is done via aligned bitexts with two target languages. A total of 223 million parallel sentences were created by combining the Europarl, United Nations, OpenSubtitles2018, Global Voices, Tanzil and Tatoeba corpora from the OPUS project (Tiedemann 2012). The loss function used in training is cross-entropy, which computes the difference between the logarithm of two probabilities: the estimate obtained from the model and the gold standard.

Relevantly to this thesis, the authors also tested how the addition of English Natural Language Inference (NLI) training data affected the embeddings. They report a better performance in the English NLI test but worse cross-language performance.

2.5.5 BERT

BERT (Devlin et al. 2019), or the Bidirectional Encoder Representations from Transformers, represents an evolution in neural network architecture for Natural Language Processing compared to LSTMs. At launch, it achieved state-of-the-art results in 11 NLP tasks, including named entity recognition, question answering and paraphrasing (Devlin et al. 2019). This model is thought out to provide a general representation of language, and as such it is able to achieve high grades in such different tasks.

This model makes use of the Transformer architecture (Vaswani et al. 2017) by stacking several transformer layers (12 in the base model and 24 in the large model). The transformer architecture innovated in two important aspects: multi-headed self-attention and positional encoding. In NLP, attention is a mechanism that allows for modelling dependencies within a sentence regardless of their distance (Vaswani et al. 2017). The Transformer is an attention-based model, replacing recurrence or convolution; by using many attention heads, it allows language models to track different kinds of dependencies between the tokens.

Another important aspect of the BERT architecture is the split between *pre-training* and *fine-tuning*: the language models are trained in such a fashion that it is possible to adapt them to specific tasks afterwards. The goal of the pre-training phase is to create models capable of tracking general linguistic information for different NLP functions, while the fine-tuning adjusts them into task-specific models.

The pre-training phase consists of simple tasks tailored to its goals and architecture. On the one hand, they do not require specific pre-constructed data: the procedures are unsupervised and use current, monolingual, unlabelled – but, importantly, contiguous – text documents (the English version of BERT was trained on texts from Wikipedia and the BooksCorpus adding up to 3.3M words). On the other, the training techniques bring context awareness both within the sentences and among them. And since the Transformer architecture is not sequential, it is not limited to processing the data one token at a time (as do LSTMs), which allows for parallelism and, consequently, speed.

The general training procedure starts with the Masked Language Model task, in which 15% of the tokens in a sentence are masked and the task is to predict them, as in the so-called *cloze task*. This is analogous to word2vec's skip-gram model; however, unlike the bag-of-words approach of word2vec, in which the position of a word is not important, only its neighbours,

the positional encoding of the Transformer brings awareness about the location of the words in the sentence and in relation to each other. The second task is Next Sentence Prediction, in which the model is required to predict if sentence B follows sentence A or not. This can bring document-level awareness to the language model.

Fine-tuning is the process of adapting the general BERT model to a specific task, such as sentence classification or question answering. This means that the training of the basic models, which takes time and resources, are performed only once. The fine-tuning is much less expensive: according to the authors, the state-of-the-art results in the paper "can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model". This *transfer learning* is one of the key reasons for BERT's success in different kinds of NLP tasks. However, BERT is not engineered to yield semantically meaningful embeddings: its usage for the specific task performed in this thesis, especially without any fine-tuning, is an extrapolation of its architecture and outside its main purpose.

BERT uses a WordPiece vocabulary of 30.000 tokens. This is a similar mechanism to the ones used by fastText and LASER, in which unknown words are split into recognisable chunks. Since the attention mechanism requires all word tokens to be matched with all others, the memory and runtime requirements for BERT are quadratic. Thus, the input texts are limited to a maximum of 512 word pieces.

2.5.5.1 FinBERT

The Google AI Language team also released a multilingual version of BERT, covering a total of 104 languages (including Finnish). However, the TurkuNLP Group from the University of Turku released their own from-scratch version of the BERT language model, called FinBERT (Virtanen et al. 2019). According to them, the language-specific version performed better on all fine-tuned tasks (part-of-speech tagging, named entity recognition, dependency parsing, text classification and probing tasks modelled to capture linguistic properties).

FinBERT was trained on three genres of text in Finnish: *news* (the Yle and the STT corpora), *online discussion* (Suomi24) and *internet crawl* (Common Crawl), which were cleaned and filtered, totalling 3.3B tokens in 234M sentences. A byte-pair encoding vocabulary of 50k tokens was used. A version of the models with lowered case and stripped accents was also released.

2.5.5.2 XLM-R

The XLM-R model (Conneau et al. 2020) was released by the Facebook AI research team in 2019 to provide cross-language text representations. It is a transformer-based model pre-trained on text in 100 languages. This number was chosen as the result of what the authors call the *curse of multilinguality*, in which adding more languages eventually leads to degraded performance on monolingual and cross-lingual benchmarks (Conneau et al. 2020).

The training is performed on monolingual texts from a clean CommonCrawl corpus using subword tokenisation with a common SentencePiece vocabulary of 250K tokens, which circumvents language-specific pre-processing (Reimers and Gurevych 2020). The Masked Language Modeling (Devlin et al. 2019) is performed by sampling text from all languages in accordance with the XML model in Lample and Conneau (2019) with changes for performance at scale. This fully unsupervised method obtains an average accuracy of 71.5% on zero-shot cross-lingual classification on the XNLI dataset, outperforming Artetxe and Schwenk (2019) by 1.3%. Using parallel data, the average accuracy jumps to 75.1, and further to 76.7 when using an NLI dataset machine-translated from English to the other languages.

2.5.5.3 SBERT

Sentence-BERT (Reimers and Gurevych 2019) is a modification of the BERT model, meant to derive "semantically meaningful" sentence embeddings. According to the authors, the standard practices used when obtaining BERT embeddings are computationally too expensive and do not produce satisfactory results.

SBERT uses the pre-trained BERT (or RoBERTa) models, fine-tuning it on NLI data using siamese and triplet networks – whose objective is to minimise the distance for similar sentence embeddings while maximising the distance for dissimilar ones – in order to produce semantically more meaningful embeddings.

The same authors have also published a method (Reimers and Gurevych 2020) that extends sentence embedding models to new languages. It requires a ready-made "teacher" model in a language x and parallel sentences in this language x and one – or multiple – target languages. The new ("student") model is trained on both the original language and a translated one using mean squared loss, so that its embeddings are aligned to those of the teacher model.

The paper uses BERT as teacher model and XLM-R as student, and fine-tunes the model on English NLI and STS data.

Chapter 3

Related Work

According to Zhang, Lertvittayakumjorn, and Guo (2019), three main types of semantic knowledge are used in zero-shot learning: semantic attributes (colour, behaviour); concept ontology (e.g., knowledge graphs), which relates classes and their features; and semantic word embeddings. The latter are used extensively in this work, while concept ontologies make a brief appearance. Regarding zero-shot learning main methods, Wang et al. (2019) describe two types: classifier-based and instance-based. The former learns classifiers directly from the available data, while the latter aims to "obtain labeled instances of the unseen classes and learn classifiers from them" (2019, 25). The work on this thesis situates itself mostly on the outskirts this classification, since it does not learn classifiers.

Indeed, entirely unsupervised zero-shot classification is not a common research subject. This means that only a small pool of papers can be considered direct precursors to the present thesis. The following authors use methods that require no ready-made training data:

- Dauphin et al. (2013) train a classifier on search engine click data (queries and their subsequent clicks) to build a vector based on semantic features for the classes. Another classifier is used to calculate the best class using a distance measure. They suggest that minimising the overlap of the semantic categories (via their conditional entropy) produces a better classifier, since the resulting embeddings are farther away from each other.
- Sappadla et al. (2016) use only semantic similarity between a label and the given documents, without any training step, for multi-label classification. They use skip-gram word

embeddings to compute semantic similarity via cosine similarity. This thesis builds on these same methods, but goes beyond it with new kinds of embeddings, other similarity measures and alternative methods for building the vector space.

- Yin, Hay, and Roth (2019) describe 0SHOT-TC, an algorithm designed to bypass what they consider frequent problems in zero-shot studies: restrictive use (mainly topic categorisation), preconditions (the existence of training classes), the lack of semantic understanding of the labels, and the lack of standardised datasets and evaluations. 0SHOT-TC treats zero-shot as a textual entailment problem, answering questions like "this text is about ?" with class label candidates. This *entailment* approach is also tested in the present work.

As discussed before, most of the work in the field of zero-shot classification is geared towards the so-called zero-shot *learning*, which aims to create methods to transfer knowledge from known classes to unknown ones. So while these methods do not require the *specific* set of labels used in the classification task, labelled datasets are still required. Transfer learning from existing classes to new ones is usually done with neural networks, which facilitates the cross-over between the fields of computer vision – where most of the research is carried out – and Natural Language Processing. While the present work does not utilise neural networks directly, some of the existing research is highlighted in this section. These are meant as a survey of the possibilities within zero-shot classification, not as direct precursors to this thesis:

- Yogatama et al. (2017) use a generative LSTM (Long Short-Term Memory, a type of neural network) to assign labels to texts. The label pool is created during training, and the assignment is based on maximised joint probabilities of labels and texts. The system uses the generative abilities of the model to predict if the documents belong to an unseen class. Its accuracy drops remarkably when two hidden classes are used instead of one.

- Pushp and Srivastava (2017) concatenate word embeddings with their label embedding, pass them through a LSTM and take the last hidden state; or, alternatively, pass the word embeddings through the LSTM and take the concatenation of the resulting hidden state to the label embedding. The results are then used to classify if sentence and label are related.

- Romera-Paredes and Torr (2017) describe an "embarrassingly simple" approach that builds a semantic attribute matrix out of the training classes, which are represented

as a list of those attributes. New classes are described in similar terms, and a domain adaptation algorithm is used to carry out the knowledge transfer.

- Rios and Kavuluru (2018) build a four-step multi-label classification framework with Graph CNNs (Convolutional Neural Networks). The descriptions/glosses for each label are transformed into vectors via averaging of their embeddings, used as attention vectors for the documents and passed through a CNN that incorporates information from a hierarchical structure and then matched to the document vectors. Both label descriptions and hierarchical information must be provided for the zero-shot classifier to work.
- Rabut, Fajardo, and Medina (2019) train a Word2Vec model using part-of-speech tags assigned to each word.
- Puri and Catanzaro (2019) train GPT-2, which is a so-called *generative model* – it models language in a way that makes it capable of generating texts –, to answers questions about the input texts. For example, they ask which of X categories best describe the text.
- Zhang, Lertvittayakumjorn, and Guo (2019) propose a two-phase framework with data and feature augmentation. They used class descriptions (alongside the labels), class hierarchy and the ConceptNet general knowledge graph. One CNN classifier is created for each seen class. Topic translation is performed via word embedding analogy from a seen to an unseen class: $seen\_class : word :: unseen\_class : ?$. This zero-shot classifier is binary and uses a sigmoid output.
- Ye et al. (2020) developes a semi-supervised framework to automatically select the best predictions the system makes and use them in reinforced self-training. They claim that integration of unlabelled data is usually not done in zero-shot learning contexts, and report significant improvements over BERT-only with this framework.

Chapter 4

Methods

4.1 Data

Two different custom datasets were used in this thesis. This section describes how they are built.

4.1.1 Finlex

The texts used in this dataset are sections of the Finnish Legislation as presented by the Semantic Finlex portal[1] (Oksanen et al. 2019), a Linked Open Data service that also includes some instances of Finnish jurisdiction (Supreme Court and Supreme Administrative Court).

Finnish Legislation is divided hierarchically, so that each Statute can be progressively decomposed into lower levels (Parts, Chapters, Sections, Subsections, Paragraphs, Subparagraphs). Legislation may also be amended or invalidated; the Semantic Finlex portal keeps track of these changes and allows for downloading only legislation in force, using SPARQL queries.

Sections were chosen as the standard level of representation for legislative texts because they are probably the best carriers of semantic meaning: they sit in between the broad and long Chapters and the brief and specific Paragraphs. However, Sections do not follow a specific size pattern: their average length in characters for the whole dataset (containing sections up to July 11th 2019) is 894 with a standard deviation of 763.

Since there is no labelled version of these legislative texts, a small sample of the database was

[1] https://data.finlex.fi

tagged by me as a gold standard for this thesis. A total of 100 random law sections were labelled according the following categories (all the multi-word labels consist of independent words chained together. The number in parentheses indicate the number of instances in this database):

- *asuminen kiinteistö* 'housing, real state' (9)
- *ihmisoikeudet perusoikeudet* 'human rights, basic rights' (3)
- *omaisuus kaupankäynti kuluttajansuoja* 'property, commerce, consumer protection' (9)
- *julkishallinto valtiohallinto* 'public administration' (7)
- *rahoitus* 'finance' (8)
- *verotus* 'taxation' (5)
- *yritykset yhteisöt työelämä* 'business, organizations, working life' (14)
- *liikenne kuljetus* 'traffic' 'transportation' (14)
- *perheoikeus perintöoikeus* 'family law, inheritance' (8)
- *rikosasiat oikeudenkäynti* 'crime, legal proceedings' (14)
- *koulutus* 'education' (2)
- *ympäristö* 'environment' (7)

The labelling is tentative, due to the complex nature of legislation – which often relate to different categories in a single section –, my lack of training in legal matters, and the fact that Finnish is not my mother tongue. As a result, many sections were skipped while building this dataset: those touching mainly themes outside the list of categories, deemed too short – lacking in semantic information – or too complex to have one single category assigned to them. The outcome is a simplified catalogue of law sections that cannot necessarily be taken as a faithful representation of the legislation.

Moreover, a sample of 100 items is not large enough to draw firm conclusions from. This dataset is presented as a small window into the behaviour of the algorithms for this kind of documents.

This dataset is available for download at this project's repository. The list its document URIs, alongside their assigned labels, can also be found in Appendix B.

4.1.2 Yle

There is an acute lack of openly available labelled datasets in Finnish, so an adequate text collection had to be created from scratch. This was achieved using material from Finland's national broadcasting company, Yle (Yleisradio Oy), via their Articles API, which offers written news articles alongside their metadata; access to it demands a personal token granted by Yle. The intention in creating this dataset was to produce a labelled collection of texts in which the tags are short, preferably composed of one word only, and semantically expressive.

A total of 11 different topics were chosen to be featured in the dataset, most of which can be found on Yle's homepage menu as main topics: *politiikka* 'politics', *talous* 'economy', *kulttuuri* 'culture', *luonto* 'nature', *tiede* 'science', *terveys* 'health', *liikenne* 'transportation', *urheilu* 'sports', *sää* 'weather'; additionally, the items *parisuhde* 'intimate relationships' and *rikokset* 'crimes' were chosen despite not appearing as main topics. These words were manually chosen to represent different classes of news. This manual triage was necessary because even the articles' main tags vary from broad concepts (such as the categories mentioned above) to proper names ("Stanley Cup", "Tarja Halonen") and specific concepts such as *puurakennukset* 'wooden buildings' or '#Metoo': only a small subset of them was deemed fit for general document categorisation.

Once the categories were chosen, two different datasets were built, separated by a date (01.01.2015, which has no specific meaning). The first dataset contains around 300 news articles for each category and a total of 3140 articles, and the second one around 200 per category and a total of 1960. Both are slightly imbalanced, since not all categories yield enough entries. They were built using the Yle API, sorted by relevance, using the `primary_subject` tag as target, which corresponds to the main topic of the article. Moreover, corresponding datasets containing (1) only the headlines and (2) the headlines and leads for each article were also built.

Since no statistical analysis is being performed in this work, splitting the data into two sets allows for checking if the scores obtained are at least somewhat consistent. Although the data is very similar in both, this gives more assurance that the scores are not obtained by mere chance. A small-scale test using this split is also performed (see 5.8).

This dataset cannot be redistributed. A list of identifiers for all the texts used in this dataset can be found in Appendix B.

4.2 Models

This chapter describes how the different vector spaces are created and which methods are used for handling them. The overall architecture of this project consists of four elements: a text extractor, a category retriever, a language model, and a classifier. This opens up alternative paths for exploration of the data, since most of these components are modularly built and can integrate freely with the other layers: a text represented as keywords may be paired up with a semantic enhancement category retriever, or a text summary can be classifier via semantic entailment. More details about each of these component types are given in their respective sections.

Figure 4.1 summarises the general architecture of these experiments.

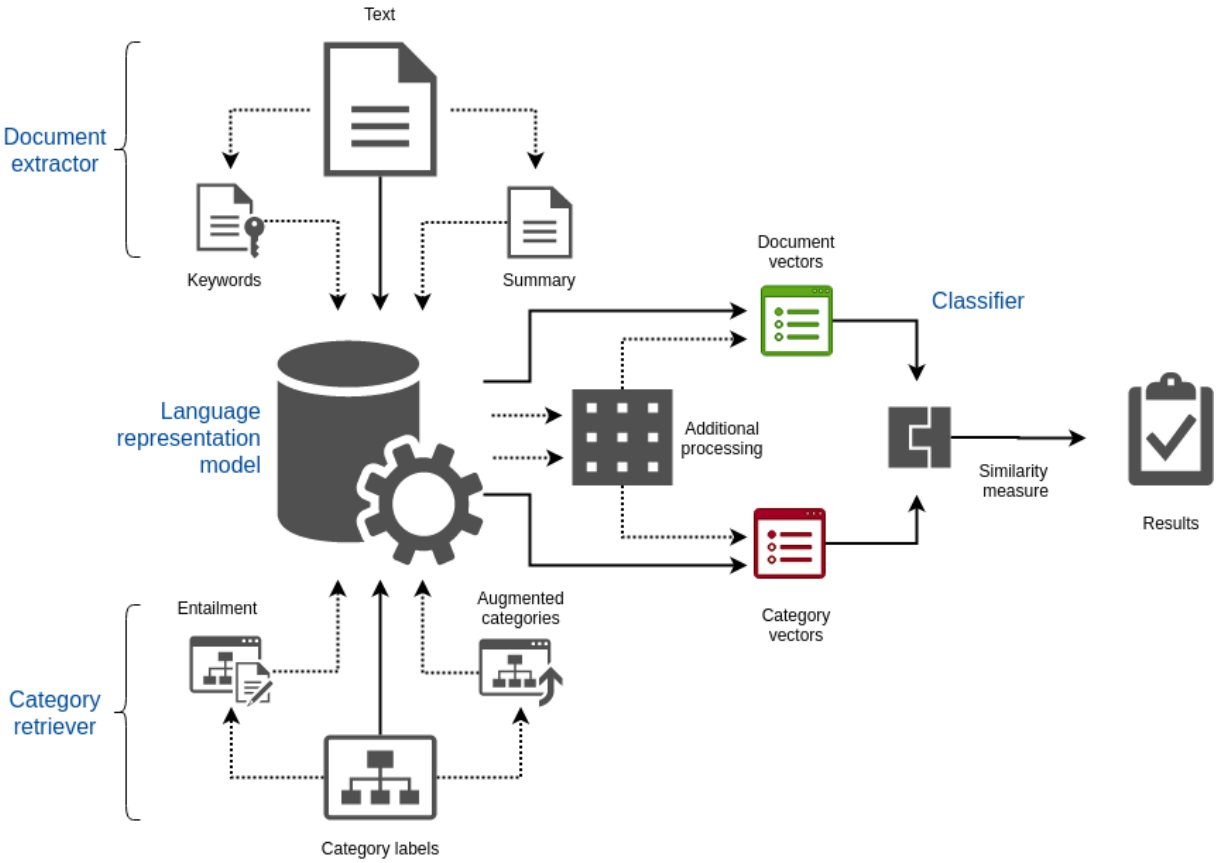


Figure 4.1: General architecture. Dotted arrows indicate alternative pathways

4.2.1 Document extractors

Extractors are responsible for representing the text. Besides the default full-text representation, an extractor may for example transform a text into keywords or into its summary.

Standard Yields the whole text of the document without any processing.

TF-IDF This is the only algorithm that does not require pre-training – but it does require text pre-processing. The documents are tokenised, split into sentences and lemmatised using the *Finnish dependency parser pipeline* by the TurkuNLP research group (Pyysalo et al. 2015). This extractor yields a list of terms and their respective weights for each document.

Annif The Annif team has released a Python command-line tool, called *Annif-client*, that accesses the Annif REST API and returns subject suggestions for the input text, alongside their weights. There are different pre-trained projects to choose from. Two different projects are used in this work: Parabel-only and an ensemble TFIDF-Parabel-Maui with equal weights. This extractor represents the documents as a list of keywords. Alternatively, it may also yield their weights, but this feature is used only when the whole vector space model is Annif-based.

Summarisation Miller (2019b) presents a simple extractive text summarisation pipeline, which obtains sentence embeddings from BERT and selects the closest representations to K centroids as candidate summary sentences. The selection is made via K-Means, which re-calculates the position of the centroids via Euclidean distance until a balance is reached. Euclidean distance is a close relative to Euclidean norm (equation 2.10) and has the following formula:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2} \tag{4.1}$$

for any two vectors in R^n. The chosen sentences are thus the ones closest to the K "centers of mass" of the vectorial representations in the Euclidean space, and are used as representatives for the whole text. The implementation used, available on GitHub, was done by the author of the paper (Miller 2019a). In it, is is possible to specify a number of sentences or a ratio for summarisation. The default of 0.2 sentences per document is used in this thesis. This tool can also directly return the embeddings obtained from the specified BERT model.

The intention behind using this method is to find out if summaries are better suited than multi-paragraph texts for zero-shot classification using a space vector model. This

could be the case if their representation lie closer to the single word embedding than the whole-text representation.

4.2.2 Language representation models

The language representation models are responsible for creating a vectorial representation for each document in the collection. They accept a text representation of the document or class and transform them into a vector in the vector space model, so that all objects have comparable representations. Models that are geared towards word representations – fastText, for example – might not be able to represent sentences effectively, let alone multi-paragraph text documents. Two main pooling strategies are regularly used in research in order to transform word vectors into sentence vectors without changing their dimensions: in *mean pooling*, the word vectors in the sentence are averaged element-wise. For all vector representations $v_0 \dots v_N$ and all vector features $v^{s0} \dots v^{sK}$:

$$v_{meanpooling} = [mean(v_0^{s0}, \dots, v_N^{s0}), mean(v_0^{s1}, \dots, v_N^{s1}), \dots, mean(v_0^{sK}, \dots, v_N^{sK})] \quad (4.2)$$

While in *max pooling*, the maximum value for each component is chosen out of all word vectors:

$$v_{maxpooling} = [max(v_0^{s0}, \dots, v_N^{s0}), max(v_0^{s1}, \dots, v_N^{s1}), \dots, max(v_0^{sK}, \dots, v_N^{sK})] \quad (4.3)$$

The intuition behind the former is that the average of the word vectors will bring about a mean vector that is a good representation of the text; for the latter, that representation will be the sum of its most salient features.

Vectorial representation for multi-sentence documents are obtained in a similar fashion: sentence vectors are pooled together to generate a final depiction of the contents of the text.

TF-IDF The vector representations for a document consist of its TF-IDF score for each keyword, which is, as a rule of thumb, a very sparse representation of the data, since a document only contains a fraction of the total number of keywords in the vector space.

fastText The fastText package (Grave et al. 2018) offers a command for acquiring sentence

vectors. It works by averaging the word vectors in the sentence after the vectors are divided by their l2 norm (see equation 2.10). In this thesis, the text documents are split into sentences via the *nltk* python package (Bird, Klein, and Loper 2009).

Annif In order to build this vector space, all the documents in the dataset are first processed using the Annif extractor. Next, a matrix is built in which each keyword found by the model constitutes one dimension, and the corresponding Annif weights are their values. In other words, a matrix is built in which the rows correspond to the documents and the columns are the different keywords.

The vector space built using this technique is similar to the counter matrices used in TF-IDF, using weights instead of counts. Different parameters can be set: a minimum weight threshold, a limit in the number of results per text – only the top X results are considered, which can make each vector roughly equal in the number of non-negative values – and a project (for example *Parabel*-only, or an ensemble with *Parabel* + *Maui* + *TF-IDF*, called *yso-fi*). The defaults are a threshold of 0.01, a limit of 100 and the previously mentioned ensemble.

As with TF-IDF, the matrices created by this model are usually very sparse. The default settings yield around 2 to 3 values per every 1000 entries.

LASER The corresponding vector space model is created using the unofficial *laserembeddings* package (yannvgn 2019).

BERT Two versions of FinBERT are used in this project: one without modifications, and another with a further round of pre-training on a larger Finlex dataset containing 65k law sections. The parameters used for fine-tuning are the recommended ones: `max_seq_len` of 128, `masked_lm_prob` of 0.15, `num_train_steps` of 10k and a learning rate of 2e-5.

The internal representation of FinBERT's models were obtained via the bert-as-a-service project (Xiao 2018), which is in its essence an API for extracting embeddings from BERT (and derivative) models. It encodes text into a fixed-length vector: padding and truncation are used to adjust the length of the text so that the resulting length is fixed.

The type of pooling strategy can be set as a parameter. Besides mean and max, it is possible to get a concatenation of these two or the hidden state corresponding to the `[CLS]` or `[SEP]` tokens (respectively first and last tokens, which contain information about

the whole sentence. These are also used in research). In this work, the embeddings for single-word labels are by default obtained using the pooling strategy `NONE`, which yields a vector for the word (or sentence) queried, without any pooling; text representations are obtained using `NONE` or `MEAN`. The `max_seq_len` parameter is set to 45 instead of the default 25, in a bid to balance speed and efficacy.

It is also possible to extract embeddings from different layers of the model, but the tests reported in Devlin et al. (2019) show that using the output of the second-to-last layer is a good balance between effectiveness and performance, and this is the default in bert-as-a-service. The best-performing strategy reported in the paper is the concatenation of the four last layers, resulting in 3072-dimensional vectors (up from the default 768), but for an F_1 score gain of only 0.4%.

SBERT Some models were also created with Sentence-BERT, which provides its own API for obtaining embeddings. The scripts for the training were adapted from the sentence-transformers GitHub page[2] otherwise using the standard parameters:

- An English-Finnish model with XLM-R base as student, and 300k parallel sentences (EN-FI) from the Europarl, EUbookshop and TildeMODEL datasets in the OPUS collection (Tiedemann 2012). The standard parameters were used: 20 epochs, 1000 evaluation steps, and a learning rate of 2e-5. (S-BERT XLMR)
- The FinBERT fine-tuned model trained on the English NLI dataset. (S-BERT NLI)

4.2.3 Post-processing and composite mechanisms

These mechanisms combine different language models or process the available representation after they are created:

TruncatedSVD This technique can be used with any of the language models. It is initially applied over the matrix representing the documents, which yields a model that can then be applied to any vector or matrix with the same number of dimensions in R – the category matrix in this case. Both matrices are then compared following normal procedures. These models are created using scikit-learn (Pedregosa et al. 2011).

Projection Following Davison (2020), this classifier takes the K most common words in the

[2] https://github.com/UKPLab/sentence-transformers

documents and learns a least-squares linear projection Z from a sentence-oriented word embedding model onto a word-oriented model, for example from BERT onto fastText, with L2 regularisation. This projection is then used to transform the sequence-oriented embeddings for both text sequences and class labels.

This projection model works as a kind of dimensionality reduction that "makes the label embeddings much better aligned with their corresponding data clusters" (Davison 2020). This projection theoretically adapts sentence-oriented embeddings to a more word-oriented space, resulting in possible improvement of the overall model. This is implemented using the Ridge model availale in scikit-learn.

Ensemble models These models pool together the results of two or more classifiers. Their classification matrices are first normalised, so that the document vectors, which contain the predictions given to every category for each document, sum up individually to one; the matrices are then averaged together and the result is a new classification matrix that can be scored without any further processing.

4.2.4 Category retrievers

Category retrieval mechanisms are responsible for textually representing the categories. In this classification task, each dataset is paired with a set of categories given as input. Some retrievers use semantic enhancement, transforming each category label into a list of related words.

Standard This mechanism simply yields the category labels as-is. E.g.: in the case of the Yle dataset, the labels are 'politics', 'economy', 'culture', etc (in their original Finnish form: see 4.1.2).

Annif The Annif retriever retrieves similar words by feeding the class label into the Annif API with a default threshold of 0.01 and a limit of 100 words. For example, for the label *politiikka* 'politics', the first elements returned are, besides the label itself, *poliittiset järjestelmät* 'political organisations', *politiikantutkimus* 'political studies', *poliitikot* 'politicians', *poliittinen osallistuminen* 'political participation', *valtiofilosofia* 'political philosophy', *valta* 'power, authority', etc.

Each element of the list receives a vectorial representation using the language representation model; these vectors are then pool them together (using mean pooling by default), resulting

in a vectorial representation for the category.

Finto For each class identifier, this classifier retrieves broader, narrower and related keywords from the KOKO ontology (see Linked Data). For example, the narrower results for the 'politics' label include *ammattiyhdistyspolitiikka* 'trade union politics', *biopolitiikka* 'biopolitics' and *data politiikka* 'data policy', while its related concepts are, among others, *anarkismi* 'anarchy', *ideologiat* 'ideologies' and *kansallisuusaate* 'nationalist ideology'.

These concepts are then returned as a list of words, which are processed analogously to the ones yielded by the Annif retriever.

Entailment The semantic entailment approach discussed in Ye et al. (2020) is tested in this thesis. For each class, the retrieval mechanism obtains representations for the sentence "Tämän tekstin aihe on {class}" 'The subject of this text is {class}', in which the token {class} is replaced by the class identifier. This proceeding is used with Sentence-BERT, since this model is trained on Natural Language Inference data.

An example of semantic retriever that is not tested in this work is dictionary-based: the category labels can be transformed into their dictionary definition(s), for which an embedding can be obtained.

4.2.5 Classifier

The classifier is the element that puts the previously discussed pieces together. It receives the textual representations from the category retriever and creates vectorial representations for them, using the appropriate language representation model. As explained above, in case the representation is a list of words, a pooling mechanism is used to unify them into a single category vector. All the category vectors are then stacked together to form the category matrix. The final classification is then carried out by comparing the document matrix and the category matrix via a similarity measure (see next subsection): each text is compared to each category, scores are calculated for all these pairings, and the final prediction is obtained via `argmax` (the highest-scoring category is chosen for each document being classified).

4.2.5.1 Similarity measures

These are the similarity algorithms used for comparing documents and categories. Their inner workings are explained in the 2.2 Similarity measures section.

- Dot product
- Cosine similarity
- Pearson's correlation

Chapter 5

Results and Discussion

This section reports on how the different models fare when effectively faced with the task of categorising documents. The tests are described and their results analysed.

The scores given here are calculated as the harmonic mean between Precision and Recall, also known as the F_1 score. Precision only takes into account the answers effectively given by the model, and calculates the percentage of right answers out of them; Recall looks at all *right* answers, globally, and measures the answers given by the model against them. It is theoretically easy to obtain a high Precision score by setting a high threshold, so that the model only gives results it is very confident about. That is why the F_1 measure is used: it penalises large disparities between its two components, enforcing balanced results and avoiding domination by one of the factors over the other, which may occur when using a simple average.

The harmonic mean is calculated as per equation 5.1:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.1}$$

Almost all F_1 scores presented here are nevertheless equal to their Precision and Recall components, which happens because the models outputs a result for every pairing of document and class label, so the "answers effectively given by the model" are the same as the total possible answers. As a result, by default Precision and Recall are not mentioned in this discussion. The figures given are thus F_1 scores unless otherwise noted.

The default category representation is the normal textual label and the default similarity mea-

sure is cosine similarity. The Annif language model and the Annif document extractor use by default the *yso-fi* ensemble with a threshold of 0.01 and a limit of 100; the TF-IDF model uses the corresponding TF-IDF extractor (with tokenized versions of the datasets, which yield much better results). The other models use the full text of the documents, otherwise noted. The default S-BERT model is XLM-R and the default FinBERT uses the 'none' pooling strategy for both documents and categories. *SVD* refers to the Truncated SVD model. The names of the datasets are at times shortened to Y1, Y2 and FL for Yle 1, Yle 2 and Finlex respectively.

5.1 Basic language models

This subsection is an overview of the basic results, which use default values, default document extractors, and no enhancement for the category labels. Only the internal parameters of the different models are tested.

Table 5.1: Basic results for all datasets

	Yle 1	Yle 2	Finlex
Annif	0.504	0.458	0.47
TF-IDF	0.354	0.307	0.2
LASER	0.104	0.102	0.09
fastText	0.63	0.649	0.45
FinBERT	0.492	0.503	0.38
S-BERT	0.417	0.401	0.28

The main tendencies are already visible in table 5.1: LASER performs much worse than the TF-IDF baseline, and indeed obtains results very close to random (around 0.09 for the Yle datasets and 0.08 for Finlex). From these results, it is already clear that this model is unsuitable for unsupervised text classification. FastText is the best-performing out-of-the-box model.

The Finlex dataset proves to be a greater challenge than the Yle-based ones for these algorithms. As discussed in the Data section, this may be due to the nature of the dataset as well as its method of creation: the language of legislative texts is unusual, containing specific terms, and the category labels used for this dataset are sometimes composed of multiple words, which can add to the challenge. Moreover, it is possible that considering only the most relevant prediction

lowers the scores, since each section of the law usually touches upon different categories - and even in case they are all recognised, their relative rank may vary. A scorer that is able to check multiple categories per document would possibly show a somewhat brighter picture – but it would demand a much more expertly built dataset.

5.2 SVD

Table 5.2 shows the effect that SVD has in this classification task. Since the number of SVD components is an input parameter, several different figures are probed and tested independently for each model and dataset. By default, the numbers are 3, 5, 10, 15, 20, 30, 45, 70, 85, 100, 120, 150, 175, 200, 250, 300, 400, 500, 600, 720, 850 and 1000 – as long as they are lower than the original size of the model. Since this sequence of numbers leaves many gaps, the highest SVD figures presented in these sections are just rough approximations.

The figures after the slash ("/") indicate the number of components (the original size of the embeddings are 768 for BERT, 300 for fastText and 1024 for LASER; for Annif and TF-IDF, they vary according to the dataset: respectively from 510 to over 6k, and from over 2k to over 66k). The Base ("B") and "SVD" numbers are F_1 scores; "Diff" is the improvement rate of SVD over the base value.

Table 5.2: Basic SVD results

	Y1 B	Y1 SVD	Diff	Y2 B	Y2 SVD	Diff	FL B	FL SVD	Diff
Annif	0.504	0.604 / 70	+19.8%	0.458	0.616 / 45	+34.5%	0.47	0.49 / 120	+4.2%
TF-IDF	0.354	0.696 / 30	+96.2%	0.307	0.627 / 30	+104.3%	0.2	0.29 / 20	+45%
LASER	0.104	0.113 / 45	+8.2%	0.102	0.157 / 3	+54.8%	0.09	0.09 / 70	0%
fastText	0.63	0.636 / 45	+1%	0.649	0.678 / 100	+4.4%	0.45	0.58 / 70	+28.9%
FinBERT	0.492	0.58 / 200	+17.9%	0.503	0.577 / 250	+14.7%	0.38	0.44 / 85	+15.8%
S-BERT	0.417	0.427 / 120	+2.4%	0.401	0.418 / 120	+4.5%	0.28	0.29 / 70	+3.6%

SVD is capable of improving the scores of almost all models – in many cases substantially, including a boost of 91-104% to TF-IDF's capabilities on the Yle datasets, resulting in a very competitive score. The average rate of improvement over all scores is ~25%, and as a general rule it is felt more intensely with sparse models (Annif and TF-IDF), which give much more

leeway for matrix decomposition as explained in the Background section. TF-IDF, which can be considered a baseline algorithm, yields the highest results in this group for Yle 1 and the second-highest for Yle 2 when SVD is applied. This shows that matrix decomposition can help distill semantic meaning into more meaningful representations.

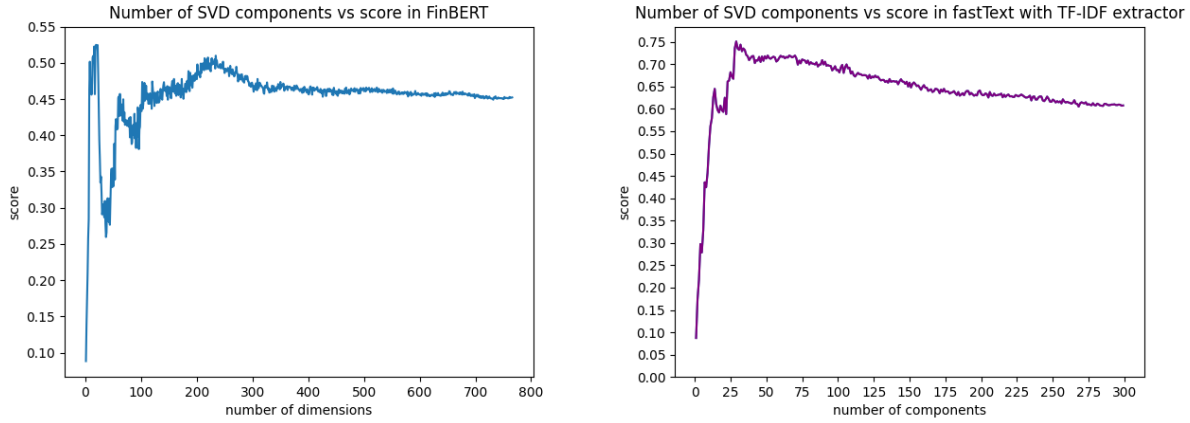


Figure 5.1: Number of SVD components vs score. FinBERT and fastText w/ TF-IDF on Yle 1

Figure 5.1 plots the scores for two language models against the number of possible SVD components (more about the second model in a subsequent section). As the number of components grow, the figures gets closer to the base score, since adding back SVD components amounts to progressively rebuilding the decomposed matrix. The "peak" at around 20 components (which is missed by the probing algorithm) in the left-most plot may be a fluke, considering the deep valley that comes right after it, but, the "plateau" is quite typical, as the right-most plot shows, and this points to a reassuring stability in SVD outcomes.

It is important to point out that SVD brings about changes that are entirely based on *existing* features, even though it acts externally and after the model's matrix has been built. There is no knowledge added, only knowledge distilled. Its consistency in improving all models in almost all settings tested grants SVD the *potential* to improved unseen data as well, and because of this its results are reported here.

However, the fact that SVD demands an input parameter (number of components) means that it cannot be used as such with unseen data: the "peaks" in figure 5.1 are not necessarily the same for different datasets. So, while non-SVD results reported here probably give a good idea of how unseen datasets would be scored using a corresponding model, the same is not necessarily true for SVD, so its results cannot be generalised. Section 5.8 tries to assess how

transferable this parameter is.

5.3 Similarity measures

Table 5.3 summarises the results obtained for different similarity measures using the basic settings. There is little difference among cosine and correlation: the mean scores are 0.369 for cosine, 0.351 for dot product and 0.369 for Pearson's correlation. When LASER is excluded, the averages are respectively 0.423, 0.405 and 0.417. Cosine similarity has a slight edge, augmented by convention and tradition. It seems that the lack of normalisation in dot product's formula prevents it from performing at the same level as the other algorithms. Their computational expense is not assessed in this thesis.

Table 5.3: Cosine similarity, dot product and Pearson's correlation as similarity measures

	Y1 Cos	Y1 Dot	Y1 Cor	Y2 Cos	Y2 Dot	Y2 Cor	FT Cos	FT Dot	FT Cor
Annif[1]	0.48	0.481	0.509	0.457	0.456	0.491	0.34	0.35	0.32
TF-IDF	0.354	0.355	0.333	0.307	0.304	0.238	0.2	0.2	0.17
LASER	0.104	0.098	0.083	0.102	0.059	0.169	0.09	0.08	0.13
fastText	0.63	0.63	0.634	0.649	0.649	0.643	0.45	0.45	0.45
FinBERT	0.492	0.389	0.491	0.503	0.36	0.503	0.38	0.27	0.38
S-BERT	0.417	0.445	0.418	0.401	0.432	0.402	0.28	0.31	0.28
Average	*0.413*	*0.4*	*0.411*	*0.403*	*0.377*	*0.408*	*0.29*	*0.277*	*0.288*

5.4 Text extraction techniques

One of the tests realised was the summarisation of the texts, according to section 4.2.1 Document extractors. The headlines and leads of the Yle datasets were also used as a summarisation proxy. The results for these tests are shown in table 5.4. Only dense, word embedding-based models were tested, which excludes TF-IDF and Annif. LASER's output is not taken into account in this analysis, since it is too close to the random baseline to be useful.

The Headlines dataset contains about 65 characters in average, and yields a mean score ap-

[1]Using Parabel and a limit of 0.01

proximately equal to 80.5/79.8% (Base/SVD) of those for the full text datasets, which contain ~2111 characters (a ratio of 3%). The figures for Headlines + Leads are: around 11% of the size (233 characters) and 90.4/89% of the scores. The summarisation technique yields 93.7/94.6% of the base results in roughly 680 characters (~32% – around 6 sentences against ~21). These numbers indicate that larger documents might be a better target than shorter ones. The gains are progressively smaller as the size of the texts grow: *diminishing returns* are a constant in Machine Learning and related NLP tasks.

Table 5.4: Headlines, Headlines + Leads and Summary: Yle 1 and Yle 2 average values

	Base Avg	Hl Avg	Hl+L	Summ	Base SVD	Hl SVD	Hl+L SVD	Summ SVD
LASER	0.103	0.111	0.11	0.108	0.135	0.112	0.112	0.121
fastText	0.64	0.509	0.599	0.617	0.657	0.542	0.641	0.643
FinBERT	0.497	0.427	0.453	0.441	0.578	0.478	0.48	0.521
S-BERT	0.409	0.308	0.354	0.392	0.423	0.314	0.365	0.405

On a similar note, but with different outcomes, another NLP text extraction technique consists in reducing documents to their representative keywords: this has been in use since the inception of TF-IDF. This technique can also be combined with word embedding algorithms: the embedding for each of the keywords for a text is pooled together to form the document representation, without the need to form sentence representations. Table 5.5 shows the results for this combination using two keyword-based text extractors: TF-IDF with a threshold of 0.7 (only the 30% globally highest-scoring words are taken into account) and Annif using the Parabel-only project with a threshold of 0.01. FinBERT is using the "no pooling" strategy: it directly extracts the vectors corresponding to the category label, without any pooling. This means that one of the phases in the "typical" BERT setup, creating sentence vectors, is skipped. In other words, it behaves more alike word2vec and fastText.

Table 5.5: Text extraction with TF-IDF at 0.7 and Annif at 0.01 compared to base values

	Y1 B	TF07 Y1	Ann Y1	Y2 B	TF07 Y2	Ann Y2	FL B	TF07 FL	Ann FL
LASER	0.104	0.096	0.364	0.102	0.102	0.352	0.09	0.1	0.41
fastText	0.63	0.575	0.735	0.649	0.606	0.727	0.45	0.4	0.64

	Y1 B	TF07 Y1	Ann Y1	Y2 B	TF07 Y2	Ann Y2	FL B	TF07 FL	Ann FL
FinBERT	0.492	0.277	0.694	0.503	0.251	0.669	0.38	0.06	0.61

Table 5.6: SVD-enhanced text extraction with TF-IDF at 0.7 and Annif at 0.01 compared to base SVD values

	Y1 SVD	TF07 Y1 SVD	Ann Y1 SVD	Y2 SVD	TF07 Y2 SVD	Ann Y2 SVD	FL SVD	TF07 FL SVD	Ann FL SVD
LASER	0.113 / 45	0.096 / 100	0.372 / 45	0.157 / 3	0.102 / 85	0.358 / 70	0.09 / 70	0.1 / 30	0.45 / 15
fastText	0.636 / 45	0.684 / 45	0.743 / 20	0.678 / 100	0.735 / 30	0.731 / 20	0.58 / 70	0.54 / 30	0.7 / 30
FinBERT	0.58 / 200	0.357 / 15	0.711 / 20	0.577 / 250	0.421 / 20	0.689 / 20	0.44 / 85	0.22 / 70	0.62 / 70

TF-IDF as a document extractor yields results that are mostly worse in comparison to the base scores, except in combination with fastText and SVD – despite the fact that with fastText alone, without SVD, the scores are lower. However, the results obtained using the Annif extractor show a very significant increase in all tests realised. The average ratio of increase over baseline is around 34%, or 117% when including LASER. Even this latter model, the worst-performing of them all, manages to achieve a score of 0.41 on the worst-performing dataset, Finlex, while FastText manages to correctly classify 64% of its items.

Thus, describing documents as a list of representative keywords and pooling these keyword's vectors together seem to be a relatively effective way to categorise texts. Annif works better than TF-IDF because it brings its own "knowledge of the world" acquired during training, which TF-IDF lacks, since it can only notice what is written in the datasets. Annif is able to extrapolate from one concept to another, filling gaps that TF-IDF cannot. And fastText is geared towards producing effective word embeddings: combining the two seems like a natural option.

5.5 Category enhancement

A logical progression in this research is to also semantically enhance the category labels. This can be done for example via the Finto or Annif category retrievers. Tables 5.7 and 5.8 show the average score for each method across all three datasets. (These tests were also carried out using max pooling for the category vectors, but mean pooling fared much better – by a margin of almost 80% in average, reduced to 20% with SVD. The results shown use mean pooling. The Annif extractor is using Parabel).

Table 5.7: Semantically enhancing the category labels on Yle 1

	TFIDF	TFIDF+Finto	Annif	Annif+Finto	Annif+Annif
LASER	0.082	0.095	0.364	0.256	0.237
fastText	0.507	0.497	0.735	0.67	0.72
FinBERT	0.199	0.274	0.694	0.51	0.54

Table 5.8: Semantically enhancing the category labels (SVD) on Yle 1

	TFIDF	TFIDF+Finto	Annif	Annif+Finto	Annif+Annif
LASER	0.096	0.14	0.372	0.259	0.234
fastText	0.606	0.564	0.742	0.70	0.739
FinBERT	0.296	0.285	0.711	0.61	0.627

The non-enhanced versions fare invariably better. It seems that the abundance of concepts confounded the models instead of helping them. Maybe a more restrictive approach, with just a handful of extra concepts, would be more appropriate, but this line of enquiry is not followed up in the present work.

5.6 Compound models

5.6.1 Projection

The projection explained in Davison (2020) (see 4.2.3) is carried out with FinBERT and fastText, so that FinBERT's vectors were mapped onto fastText's. This model scores 0.463 and 0.469 (the

latter for SVD with 50 components) for the Yle 1 dataset, which is lower than base results for the models (0.492 and 0.63).

5.6.2 Ensembles

In this section, some of the best-scoring models are brought together as ensembles. There is some improvement to be had in combining different models, as table 5.9 (Yle 1) shows on rows 3 and 4, table 5.10 (Yle 2) on its only row, and table 5.11 (Finlex) on row 2. The combination of fastText and fastText with the Annif extractor achieves the highest scores reported in this thesis for all datasets (see analysis in 5.9 for some analysis and visualisation). Ensembles containing SVD do not seem to be able to improve on its components' scores.

Table 5.9: Ensemble scores for the Yle 1 dataset

Model 1	Model 2	M1 score	M2 score	Ensemble score
fastText SVD	TF-IDF SVD	0.636	0.695	0.718
fastText w/ Annif-ext SVD	FinBERT w/ Annif-ext SVD	0.743	0.711	0.741
fastText w/ Annif-ext	FinBERT w/ Annif-ext	0.721	0.694	0.738
fastText	fastText w/ Annif-ext	0.630	0.735	0.760

Table 5.10: Ensemble scores for the Yle 2 dataset

Model 1	Model 2	M1 score	M2 score	Ensemble score
fastText	fastext w/ Annif-ext	0.649	0.731	0.763

Table 5.11: Ensemble scores for the Finlex dataset

Model 1	Model 2	M1 score	M2 score	Ensemble score
FinBERT w/ Annif-ext SVD	FastText w/ Annif-ext SVD	0.62	0.7	0.67
fastText	fastext w/ Annif-ext	0.45	0.64	0.65
fastText	fastext w/ Annif-ext SVD	0.45	0.7	0.65

5.7 Individual models

This section collects model-specific results.

5.7.1 FinBERT

Four different pooling combinations are tested alongside two different versions of FinBERT, one pre-trained and one fine-tuned. The pre-trained model fares almost invariably better, which may be due to the fact that only one fine-tuned version was produced, without further parameter-tuning. The best-performing models, once again, use the Annif extractor and SVD. Considering only BERT's internal strategies, pooling sentences with mean pooling and comparing them to category labels extracted with "no pooling", which returns embeddings akin to sentence embeddings, fares best – but only when enhanced via SVD; otherwise, no pooling for both sentences and labels is the best option. This finding shows that mean pooling, despite being the default option in *bert-as-a-service* (Xiao 2018), does not necessarily build the best semantic vectors when dealing with BERT. In any case, fastText embeddings seem to be more suitable for this categorisation task.

Table 5.12: FinBERT pre-trained model (PT) compared to its fine-tuned (FT) version on Yle 1

Pooling strategy	PT B	PT SVD	FT B	FT SVD
NONE (Annif ext)	0.577	0.64 / 20	0.535	0.57 / 30
MEAN	0.45	0.52 / 20	0.385	0.565 / 30
MEAN + NONE	0.469	0.639 / 150	0.322	0.546 / 30
NONE	0.492	0.58 / 200	0.353	0.5 / 30

Table 5.13: FinBERT pre-trained model (PT) compared to its fine-tuned (FT) version on Finlex

Pooling strategy	PT B	PT SVD	FT B	FT SVD
NONE (Annif ext)	0.61	0.62 / 70	0.535	0.574 / 20
MEAN	0.28	0.38 / 30	0.109	0.109 / 85
MEAN + NONE	0.22	0.55 / 85	0.277	0.475 / 30
NONE	0.38	0.44 / 85	0.25	0.32 / 70

5.7.2 S-BERT

S-BERT was taken up on its promise of better semantic vectors, but failed to deliver. Almost all results are worse than FinBERT's above, and the entailment approach yielded at most mixed results.

Table 5.14: S-BERT using different models

	Y1	Y2	FL	Y1 SVD	Y2 SVD	FL SVD
S-BERT	0.417	0.401	0.28	0.427 / 120	0.418 / 120	0.29 / 70
S-BERT + entailment	0.343	0.293	0.27	0.347 / 120	0.299 / 120	0.32 / 20
S-BERT NLI	0.395	0.39	0.47	0.418 / 20	0.401 / 70	0.47 / 30
S-BERT NLI + entailment	0.437	0.416	0.3	0.439 / 200	0.417 / 175	0.34 / 5

5.8 Prediction using SVD components

In table 5.15, the number of components in some SVD-enhanced models on Yle 1 is used to build corresponding models for the other datasets. The results obtained are then compared to the best predictions on those datasets, in order to assess how transferable this machine-learned parameter is from one dataset to another.

Table 5.15: Number of components in Yle 1's best SVD model used to build corresponding models for Yle 2 and Finlex

Model	Y1 SVD comp	Y2 Base	Y2 SVD Pred	Y2 SVD Best	FL Base	FL SVD Pred	FL SVD Best
Annif	70	0.458	0.566 / 70	0.616 / 45	0.47	0.43 / 70	0.49 / 120
fastText	45	0.649	0.655 / 45	0.678 / 100	0.45	0.52 / 45	0.58 / 70
fastText w/ Annif-ext	20	0.73	0.75 / 20	0.75 / 20	0.64	0.63 / 20	0.7 / 30
FinBERT[2]	150	0.446	0.609 / 150	0.637 / 100	0.22	0.55 / 150	0.55 / 85

Model	Y1 SVD comp	Y2 Base	Y2 SVD Pred	Y2 SVD Best	FL Base	FL SVD Pred	FL SVD Best
FinBERT[3] w/ Annif-ext	20	0.554	0.615 / 20	0.615 / 20	0.61	0.57 / 20	0.62 / 70

Based on this small number of data points, it seems that this approach works for Yle 2, a very similar dataset, but falters when it comes to Finlex. All the predicted results for Yle 2 are higher than the base value, and sometimes as high as the best score. In the case of Finlex, all models involving Annif as an extractor yield values lower than the baseline, while the other raise the score. This can be explained by the different plot shapes in figures 5.2 and 5.3: with Annif, the initial curve of the Yle datasets rises very steeply, leaving Finlex behind. For both fastText and FinBERT[4], the peak of the Finlex plot was missed by a very small number of components (around 10: the plots are more reliable than the tables, because a much greater numbers of components is tested in them. Base score means without SVD). Given enough data, perhaps a machine learning algorithm could make the SVD predictions more stable and thus allow them to be used on unseen data.

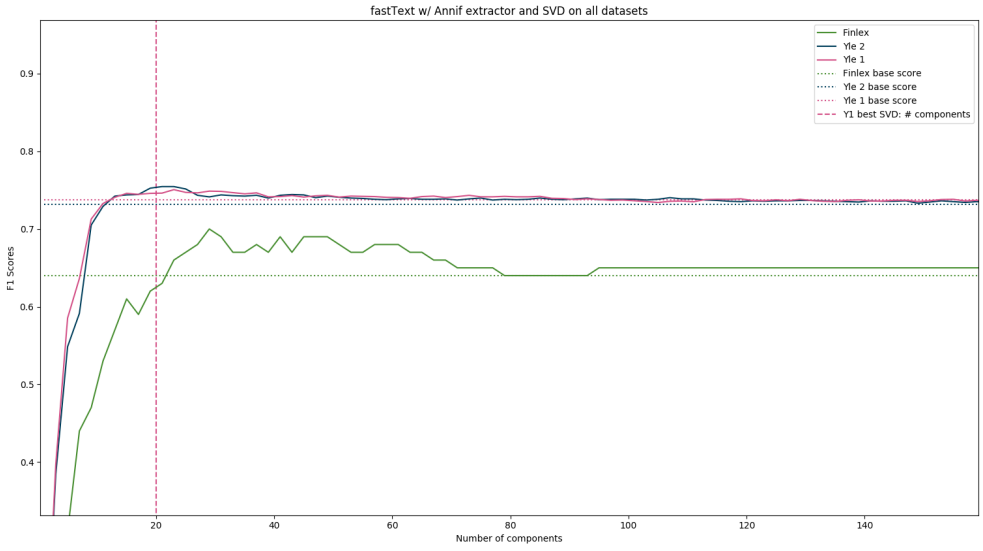


Figure 5.2: SVD components plot: fastText w/ Annif and SVD (cropped)

[2]Using mean pooling for the documents and 'none' for the categories

[3]Using mean pooling for the documents and 'none' for the categories

[4]Plots for FinBERT, which are quite similar, can be found in Appendix A, figures 6.4 and 6.5.

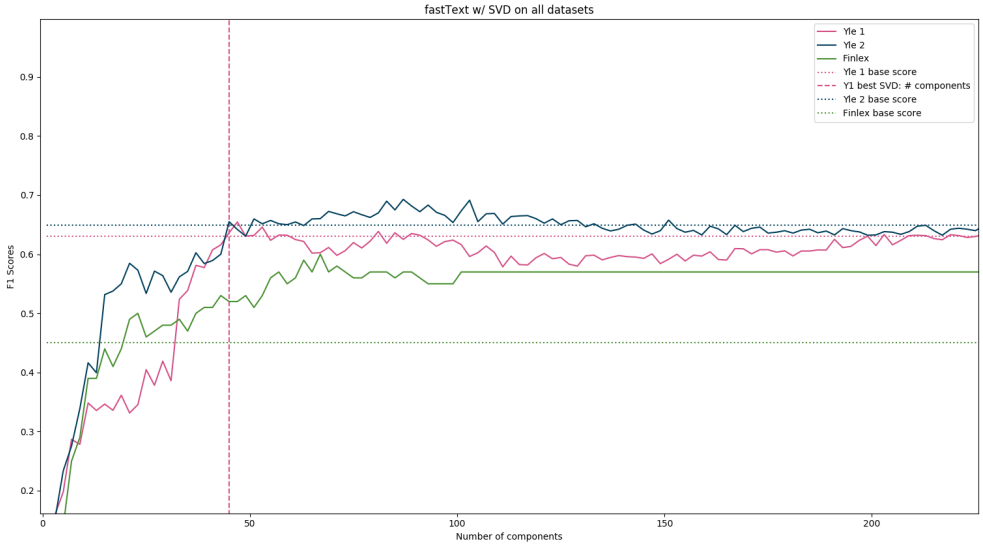


Figure 5.3: SVD components plot: fastText w/ SVD (cropped)

5.9 Error analysis

The graphs in this section, commonly called "confusion matrices", help visualise the predictions made by the models. Each row shows the distribution of predictions for a label, and each column corresponds to a different prediction. Scores in the main diagonal indicate the percentage of correct predictions. A visible vertical column means that documents belonging to different categories are erroneously being classified into one and the same category.

Figure 5.4 shows the results for fastText, by itself (F_1 = 0.63) and using Annif as document extractor (F_1 = 0.735) on Yle 1. It explains why their ensemble works: since they use different kinds of document extraction methods, part of their individual weaknesses are counterbalanced by strong numbers on the other side[5]. FastText alone is unable to identify two of the categories: *tiede* 'science' and *terveys* 'health', assigning around one third of them to *kulttuuri* 'culture': there are no significant error patterns that make semantic sense (around 16% of *talous* 'economy' is assigned to 'politics', but also as much is labelled as 'culture'). Regarding the second model, where Annif keywords are used, only 'science' is significantly problematic, with only 17% of correct predictions; this and *sää* 'weather' are the only categories below 70%. Its errors are scattered, without a clear pattern (except 19% of 'weather' being classified as 'nature', which makes sense semantically). The results for Yle 2 are very similar (see fig. 6.3 in

[5]The confusion matrix for this ensemble can be found in Appendix A, figure 6.1

Appendix A).

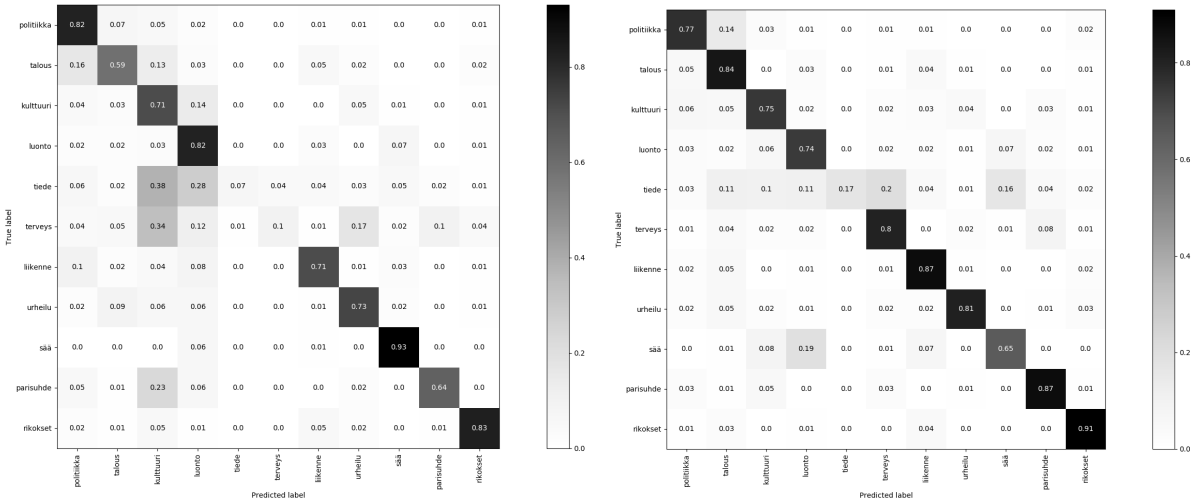


Figure 5.4: Confusion matrices: fastText by itself and using the Annif extractor (0.01) on Yle 1

Figures 5.5 and 5.6 stand in contrast. The first one shows the results for the TF-IDF model, by itself and with SVD, on Yle 1. Without matrix decomposition, the model classifies most documents as *politiikka* 'politics', although a diagonal line – mostly faint, but occasionally well-defined – can be seen. With SVD, this diagonal is well-defined. 'Science' is once again the most challenging label. LASER, as can be seen from figure 5.6, does not behave the same way with SVD: its "column", standing on *liikenne* 'traffic, transportation', gets even more marked than before (and on Yle 2, it changes place, from 'relationships' to 'sport' and 'nature'. See figure 6.2 on Appendix A). This probably means that the LASER model does not carry by itself the necessary semantic information needed for this task, while TF-IDF does. These columns mean that a category is getting a disproportionate amount of predictions, i.e., its representation is numerically too strong and drowns out other categories. This is why by itself LASER is unable to classify correctly: most of its predictions land on the same category.

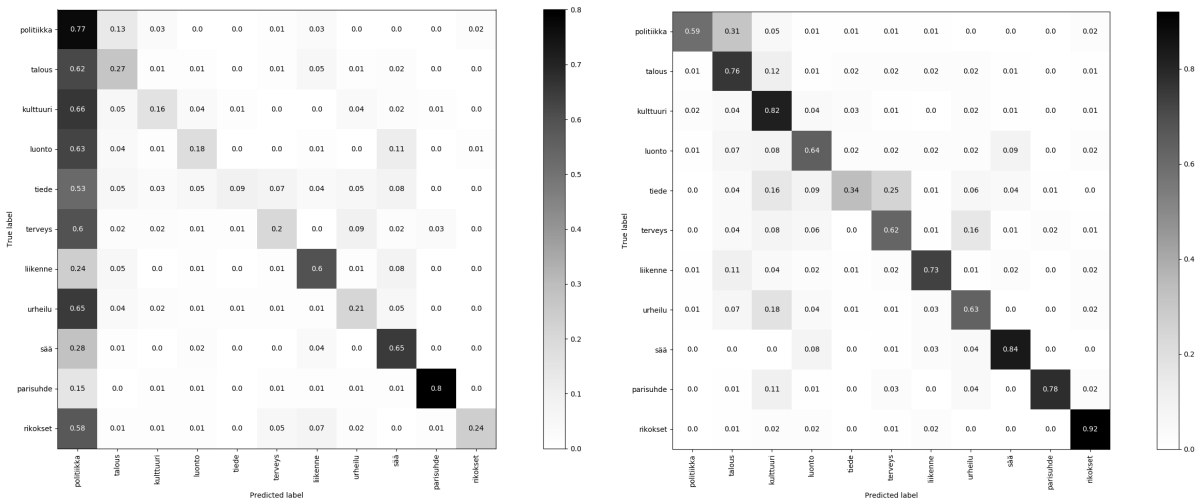


Figure 5.5: Confusion matrices: TF-IDF by itself and with 30-component SVD on Yle 1

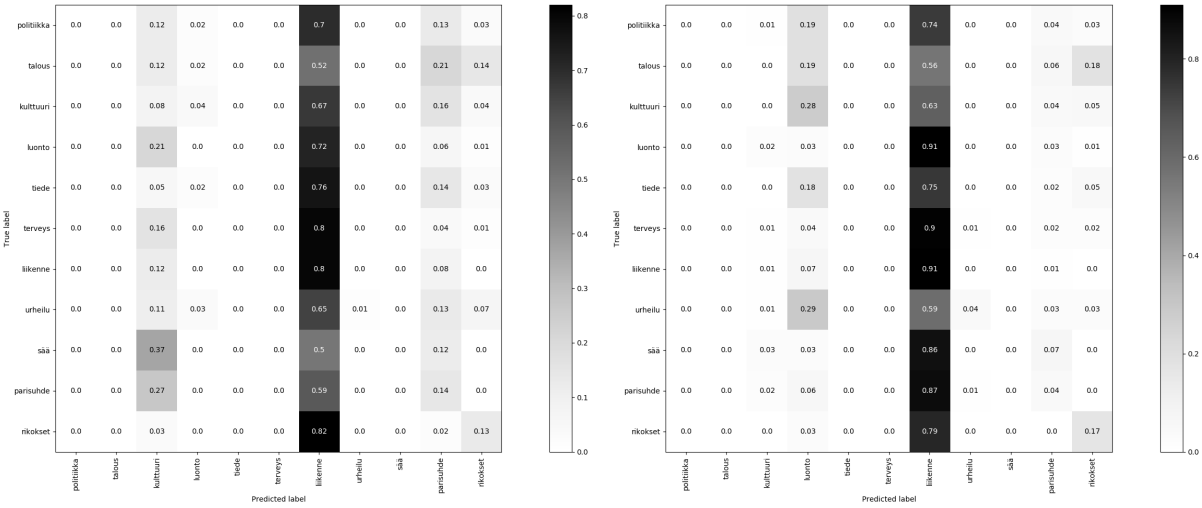


Figure 5.6: Confusion matrices: LASER by itself and with 45-component SVD on Yle 1

Finally, figure 5.7 shows the results for FinBERT (with "none" pooling) and fastText on Finlex, both with Annif extractor and SVD. The former has an F_1 score of 0.58 and the second, 0.7. This comparison aims to show how models' results differ despite having the same document extractor. The models provide their own internal representation for the words given by Annif and for the category labels. There is some overlap: faint columns over certain categories (*ihmisoikeudet perusoikeudet* 'human rights, basic rights', *julkishallinto valtiohallinto* 'public administration' and *verotus* 'taxation'), but FinBERT shows an additional, relatively consistent, column over *omaisuus kaupankäynti kuluttajansuoja* 'property, commerce, consumer protection', which is absent from fastText. This is the main cause for the differences in score between the two models.

Based on 5.7, it does not seem that multi-word labels negatively affect the predictions, but a more in-depth examination would be required in order to draw conclusions.

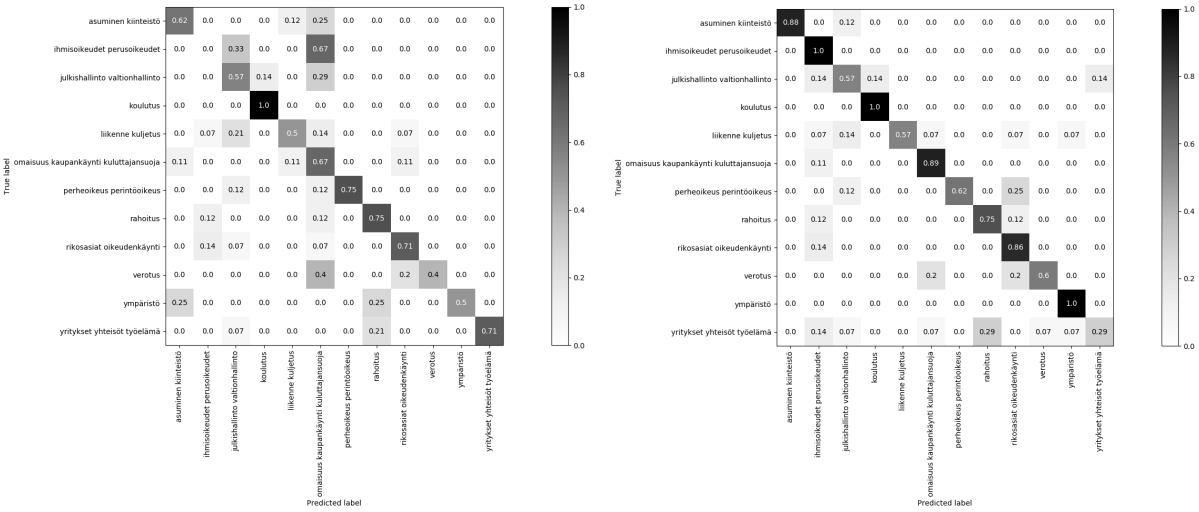


Figure 5.7: Confusion matrices: FinBERT and fastText, both w/ Annif extractor and SVD, on Finlex

In summary, these confusion matrices show two main types of errors: “rows”, which happen when the predictions for a certain label are more or less evenly spread among the targets and might be associated with a weak representation for the corresponding source label; and “columns”, when a target label gets a disproportionate amount of predictions to itself. One possible cause for this could be unbalanced training data: too few examples might produce a weaker vector and vice versa.

Chapter 6

Conclusions

The results presented in this work show that, by using pre-trained Finnish language models in an unsupervised manner, it is possible to build models able to classify texts correctly 65-76% of the time in a multi-class categorisation task. The scores can be brought up to 70-76% by using SVD, but this method is not necessarily reproducible as-is. These figures demonstrate that some of the algorithms tested carry a substantial amount of semantic information that can be drawn upon for document categorisation, even though they are not built specifically for the Finnish language.

These results – even the best ones – are in any case far from the near-perfect scores that can be obtained via supervised algorithms. Yet, unsupervised zero-shot methods offer near-instant flexibility and, above all, bypass the need for specific training data – which may not exist in large enough quantities to produce scores closer to 100%.

The best F_1 scores for the Finlex dataset are consistently lower than for the Yle datasets. This is probably due to different factors: the small number of documents in it, its inexpert construction, the kind of language it contains. A multi-label setting, which is closer to the intended real-life application for this model (in an exploratory search engine setting), would possibly yield better results by also taking into account secondary categories present in the documents – which can end up, in the eyes of these algorithms, exchanging places with the primary.

The best scores reported in this work are the result of different algorithms working in tandem: fastText provides strong baselines and Annif helps focus the efforts, and SVD *potentially* distills the knowledge obtained into a compact and highly-correlated matrix. It seems to be ad-

vantageous to pool keywords together into text representations, bypassing sentence representations. Moreover, the ensemble consisting of fastText and fastText using the Annif extractor seem to complement each other, so they raise the figures by around one to three percentage points. Other models achieve higher scores at times: even the venerable TF-IDF proved to be a challenge at a specific point, when paired with SVD. However, unlike other combinations, the best mixture provides a consistency the others lack. These results also show that it is theoretically possible to enhance the semantic payload of the basic models by using only information they already contain, and by filtering this information with the right text representation methods. Regarding similarity measures, no significant difference was found when applying cosine similarity, Pearson's correlation or dot product.

Word embeddings prove to be useful also due to their adaptability: they ultimately accept different kinds of document representations, from single words to multi-paragraph texts. Pretraining is indeed able to imbue them with enough linguistic knowledge to perform a reasonably good document categorisation. Some worked better than the others, however: LASER proved to be unsuitable for the task, and the different versions of BERT, middling. FastText, the best-performing word embedding algorithm in almost all tests reported here, is bound to be surpassed in the future, with increasingly more sophisticated algorithms being developed and more data available for training, and so is Annif. Ultimately, however, the main findings in this thesis call attention to the fact that single models may not be the best solution when it comes to unsupervised text categorisation.

Due to the sheer amount of possible combinations, many of them are left untested. It would be useful to experiment with these – different thresholds and limits for Annif, different pooling strategies and maximum sequence lengths for BERT, different ensembles – as well as other methods, such as dictionary definitions as semantic enhancements or fastText fine-tuning, in order to find possible better-performing combinations that could bring this unsupervised task closer to supervised levels. However, two main tasks stand out at this point: developing a reliable way to determine the "peak" of the SVD according to its number of components, and testing these models in a multi-label environment, instead of a multi-class one.

Bibliography

Artetxe, Mikel, and Holger Schwenk. 2019. 'Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond'. *Transactions of the Association for Computational Linguistics*. https://doi.org/10.1162/tacl_a_00288.

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc. http://nltk.org/book.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. 'Enriching Word Vectors with Subword Information'. *Transactions of the Association for Computational Linguistics* 5 (December): 135–46. https://doi.org/10.1162/tacl_a_00051.

Chai, Duo, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. 'Description Based Text Classification with Reinforcement Learning'. *ArXiv*.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. 'Unsupervised Cross-Lingual Representation Learning at Scale'. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–51. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747.

Conneau, Alexis, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. 'XNLI: Evaluating Cross-Lingual Sentence Representations'. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics.

Dauphin, Yann, Gokhan Tur, Dilek Z. Hakkani-Tur, and Larry Heck. 2013. 'Zero-Shot Learning for Semantic Utterance Classification'. *ICLR 2014*.

Davison, Joe. 2020. 'Zero-Shot Learning in Modern NLP'. Joe Davison Blog. 29 May 2020.

https://joeddav.github.io/blog/2020/05/29/ZSL.html.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 'BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding'. In *NAACL-HLT*. https://doi.org/10.18653/v1/N19-1423.

Diaz, Gene. (2016) 2016. *Stopwords-Iso/Stopwords-Fi*. Stopwords ISO. https://github.com/stopwords-iso/stopwords-fi.

Dubin, David. 2004. 'The Most Influential Paper Gerard Salton Never Wrote'. https://www.ideals.illinois.edu/handle/2142/1697.

Finley, Gregory, Stephanie Farmer, and Serguei Pakhomov. 2017. 'What Analogies Reveal About Word Vectors and Their Compositionality'. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, 1–11. Vancouver, Canada: Association for Computational Linguistics. https://doi.org/10.18653/v1/S17-1001.

'Finna.Fi'. n.d. Accessed 21 July 2020. https://www.finna.fi/.

Frosterus, Matias, Jouni Tuominen, Sini Pessala, Katri Seppälä, and Eero Hyvönen. 2013. 'Linked Open Ontology Cloud KOKO—Managing a System of Cross-Domain Lightweight Ontologies'. In *The Semantic Web: ESWC 2013 Satellite Events*, 296–97. Montpellier, France: Springer-Verlag, Berlin Heidelberg.

Gabrilovich, Evgeniy, and Shaul Markovitch. 2007. 'Computing Semantic Relatedness Using Wikipedia-Based Explicit Semantic Analysis'. In *IJCAI*.

Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. 'Learning Word Vectors for 157 Languages'. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Harris, Zellig S. 1954. 'Distributional Structure'. *WORD* 10 (2-3): 146–62. https://doi.org/10.1080/00437956.1954.11659520.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. 'Long Short-Term Memory'. *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. 'Bag of Tricks for Efficient Text Classification'. 9 August 2016. http://arxiv.org/abs/1607.01759.

Jurafsky, Dan, and James H. Martin. 2019. *Speech and Language Processing*. 3rd ed. draft. https://web.stanford.edu/~jurafsky/slp3/.

Lample, Guillaume, and Alexis Conneau. 2019. 'Cross-Lingual Language Model Pretraining'. *NeurIPS*.

Lawrence, Neil D., and John C. Platt. 2004. 'Learning to Learn with the Informative Vector Machine'. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 65. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery. https://doi.org/10.1145/1015330.1015382.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. New York: Cambridge University Press.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. 'Efficient Estimation of Word Representations in Vector Space'. 6 September 2013. http://arxiv.org/abs/1301.3781.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. 'Distributed Representations of Words and Phrases and Their Compositionality'. 16 October 2013. http://arxiv.org/abs/1310.4546.

Miller, Derek. 2019a. *Dmmiller612/Bert-Extractive-Summarizer*. https://github.com/dmmiller612/bert-extractive-summarizer.

———. 2019b. 'Leveraging BERT for Extractive Text Summarization on Lectures'. 7 June 2019. http://arxiv.org/abs/1906.04165.

Minaee, Shervin, Nal Kalchbrenner, E. Cambria, Narjes Nikzad, M. Chenaghlu, and Jianfeng Gao. 2020. 'Deep Learning Based Text Classification: A Comprehensive Review'. *ArXiv*.

Oksanen, Arttu. 2016. 'Lainsäädännön ja oikeuskäytännön mallintaminen ja julkaiseminen linkitettynä avoimena datana'. Aalto University, School of Electrical Engineering, Degree Programme in Electronics and Electrical Engineering.

Oksanen, Arttu, Jouni Tuominen, Eetu Mäkelä, Minna Tamper, Aki Hietanen, and Eero Hyvönen. 2019. 'Semantic Finlex: Transforming, Publishing, and Using Finnish Legislation and Case Law as Linked Open Data on the Web'. In *Knowledge of the Law in the Big Data Age*, edited by G. Peruginelli and S. Faro, 317:212–28. Frontiers in Artificial Intelligence and Applications. IOS Press. http://doi.org/10.3233/FAIA190023.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. 'Scikit-Learn: Machine Learning in Python'. *Journal of Machine Learning Research* 12: 2825–30.

Prabhu, Yashoteja, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. 'Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising'. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 993–1002. Lyon, France: ACM Press. https://doi.org/10.1145/3178876.3185998.

Puri, Raul, and Bryan Catanzaro. 2019. 'Zero-Shot Text Classification with Generative Language Models'. 10 December 2019. http://arxiv.org/abs/1912.10165.

Pushp, Pushpankar Kumar, and Muktabh Mayank Srivastava. 2017. 'Train Once, Test Anywhere: Zero-Shot Learning for Text Classification'. 23 December 2017. http://arxiv.org/abs/1712.05972.

Pyysalo, Sampo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. 'Universal Dependencies for Finnish'. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, 163–72. Vilnius, Lithuania: Linköping University Electronic Press, Sweden. https://www.aclweb.org/anthology/W15-1821.

Rabut, Benedict, Arnel Fajardo, and Ruji Medina. 2019. 'Multi-Class Document Classification Using Improved Word Embeddings'. In, 42–46. https://doi.org/10.1145/3366650.3366661.

Reimers, Nils, and Iryna Gurevych. 2019. 'Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks'. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–92. Hong Kong, China: Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410.

———. 2020. 'Making Monolingual Sentence Embeddings Multilingual Using Knowledge Distillation'. 21 April 2020. http://arxiv.org/abs/2004.09813.

Rios, Anthony, and Ramakanth Kavuluru. 2018. 'Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces'. *EMNLP*. https://doi.org/10.18653/v1/D18-1352.

Romera-Paredes, Bernardino, and Philip H. S. Torr. 2017. 'An Embarrassingly Sim-

ple Approach to Zero-Shot Learning'. In *Visual Attributes*, edited by Rogerio Schmidt Feris, Christoph Lampert, and Devi Parikh, 11–30. Advances in Computer Vision and Pattern Recognition. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-50077-5_2.

Salton, G., A. Wong, and C. S. Yang. 1975. 'A Vector Space Model for Automatic Indexing'. *Communications of the ACM* 18 (11): 613–20. https://doi.org/10.1145/361219.361220.

Sanderson, Grant [3Blue1Brown], dir. 2016. *Dot Products and Duality [Video]*. Vol. 9. Essence of Linear Algebra. https://www.youtube.com/watch?v=LyGKycYT2v0.

Sappadla, Prateek Veeranna, Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. 'Using Semantic Similarity for Multi-Label Zero-Shot Classification of Text Documents'. In *ESANN*.

Sarsa, Sami. 2019. 'Information retrieval with finnish case law embeddings'. University of Helsinki, Department of Computer Science.

Schwartz, Lane, Francis Tyers, Lori Levin, Christo Kirov, Patrick Littell, Chi-kiu Lo, Emily Prud'hommeaux, et al. 2020. 'Neural Polysynthetic Language Modelling'. 13 May 2020. http://arxiv.org/abs/2005.05477.

Schwenk, Holger. 2019. 'Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond [Video Recording by ACL]'. EMNLP - IJCNLP 2019, November 6. https://vimeo.com/426369997.

Shen, Dinghan, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. 'Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms'. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 440–50. Melbourne, Australia: Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1041.

Suominen, Osma. 2019. 'Annif: DIY Automated Subject Indexing Using Multiple Algorithms'. *LIBER Quarterly* 29 (1): 1–25. https://doi.org/10.18352/lq.10285.

Suominen, Osma, and Juho Inkinen. 2018. *NatLibFi/Annif-Client* (version 0.3.0). https://github.com/NatLibFi/Annif-client.

Suominen, Osma, Juho Inkinen, Tuomo Virolainen, Bruno P. Kinoshita, Sara Veldhoen, Mats Sjöberg, Moritz Fürneisen, Philipp Zumstein, Robin Neatherway, and Mona Lehtinen. 2020. *NatLibFi/Annif: Annif 0.48*. Zenodo. https://doi.org/10.5281/zenodo.3921043.

Tiedemann, Jörg. 2012. 'Parallel Data, Tools and Interfaces in OPUS'. In *In Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, 5.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. 'Attention Is All You Need'. *NIPS*.

Virtanen, Antti, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. 'Multilingual Is Not Enough: BERT for Finnish'. 15 December 2019. http://arxiv.org/abs/1912.07076.

Wang, Wei, Vincent W. Zheng, Han Yu, and Chunyan Miao. 2019. 'A Survey of Zero-Shot Learning: Settings, Methods, and Applications'. *ACM Transactions on Intelligent Systems and Technology* 10 (2): 1–37. https://doi.org/10.1145/3293318.

Xiao, Han. 2018. *Bert-as-Service*. https://github.com/hanxiao/bert-as-service.

yannvgn. 2019. *Yannvgn/Laserembeddings*. https://github.com/yannvgn/laserembeddings.

Ye, Zhiquan, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, Suhang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. 'Zero-Shot Text Classification via Reinforced Self-Training'. In *ACL*.

Yin, Wenpeng, Jamaal Hay, and Dan Roth. 2019. 'Benchmarking Zero-Shot Text Classification: Datasets, Evaluation and Entailment Approach'. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3914–23. Hong Kong, China: Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1404.

Yleisradio. n.d. 'Ylen Suomenkielinen Uutisarkisto 2011-2018, Korp'. Tekstikorpus. Kielipankki. http://urn.fi/urn:nbn:fi:lb-2019121003.

Yogatama, Dani, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. 'Generative and Discriminative Text Classification with Recurrent Neural Networks'. 25 May 2017. http://arxiv.org/abs/1703.01898.

Zhang, Ji, Yu Chen, and Yongjie Zhai. 2020. 'Zero-Shot Classification Based on Word Vector

Enhancement and Distance Metric Learning'. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.2998495.

Zhang, Jingqing, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. 'Integrating Semantic Knowledge to Tackle Zero-Shot Text Classification'. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 1031–40. Minneapolis, Minnesota: Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1108.

Zobel, Justin, and Alistair Moffat. 1998. 'Exploring the Similarity Space'. *Sigir Forum* 32: 18–34.

Appendix A

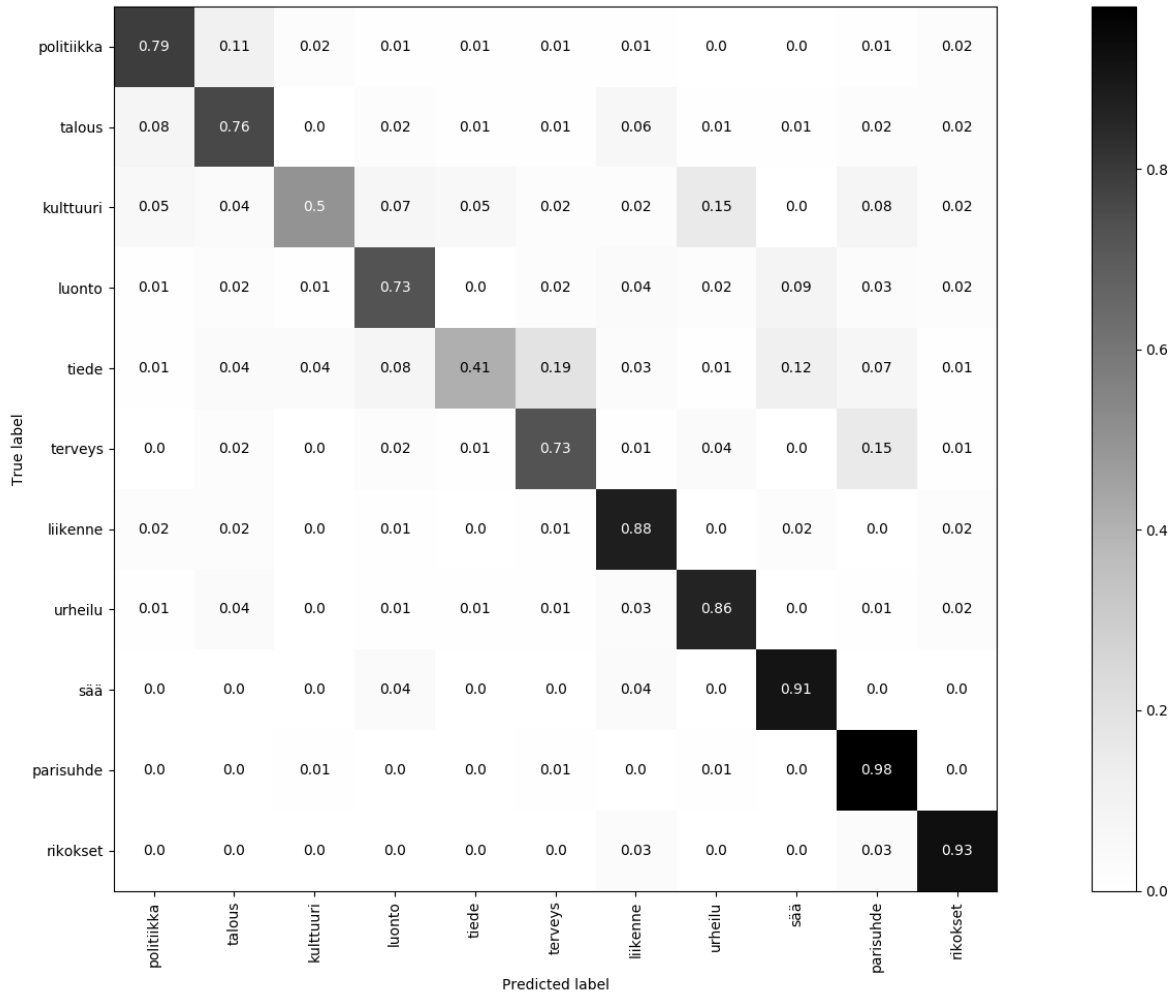


Figure 6.1: Confusion matrix: Ensemble of fastText and fastText with Annif extractor

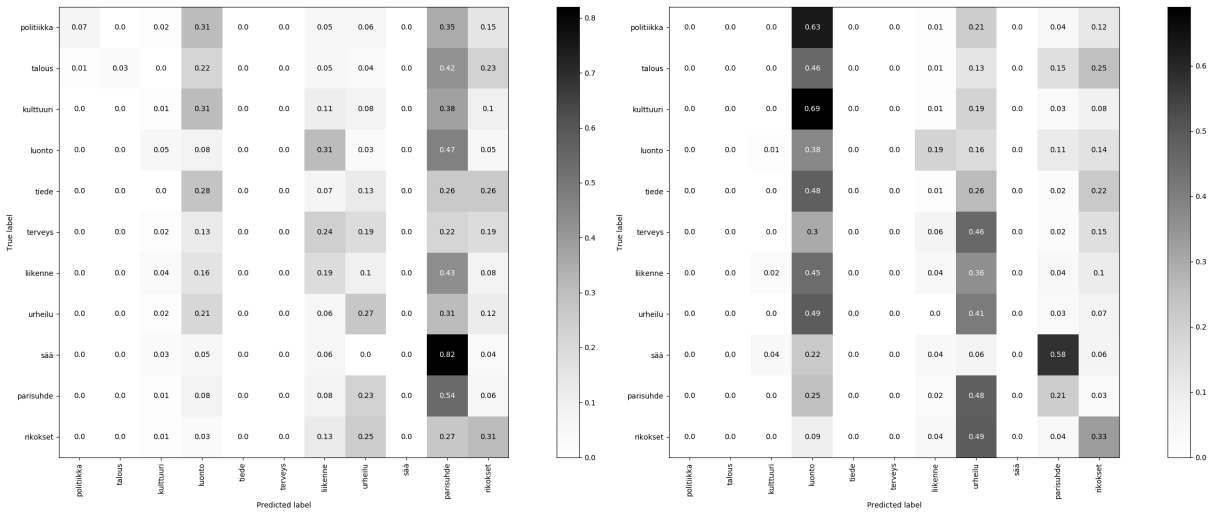


Figure 6.2: Confusion matrices: LASER by itself and with 45-component SVD on Yle 2

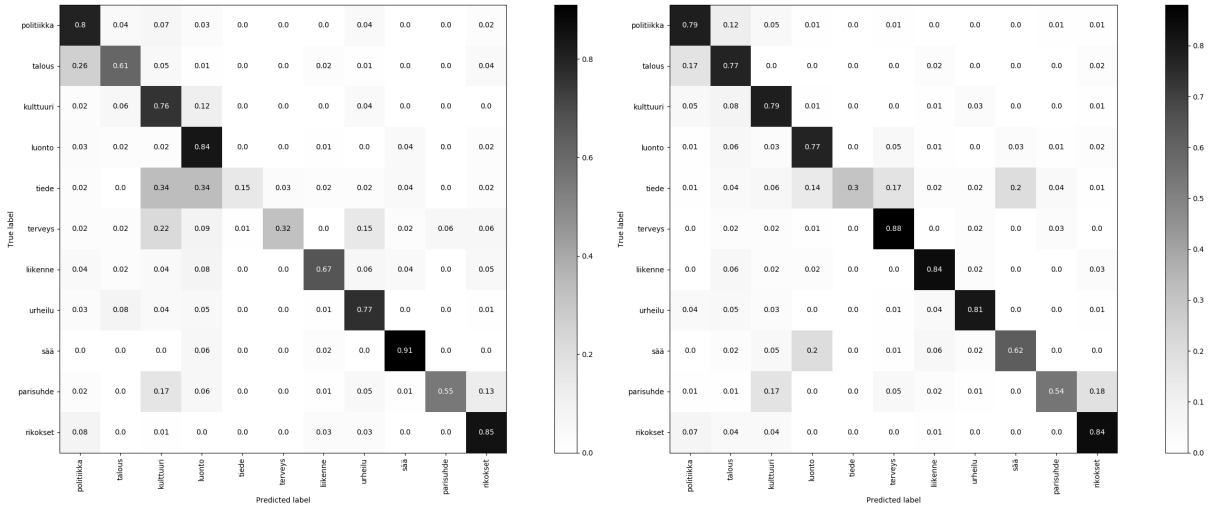
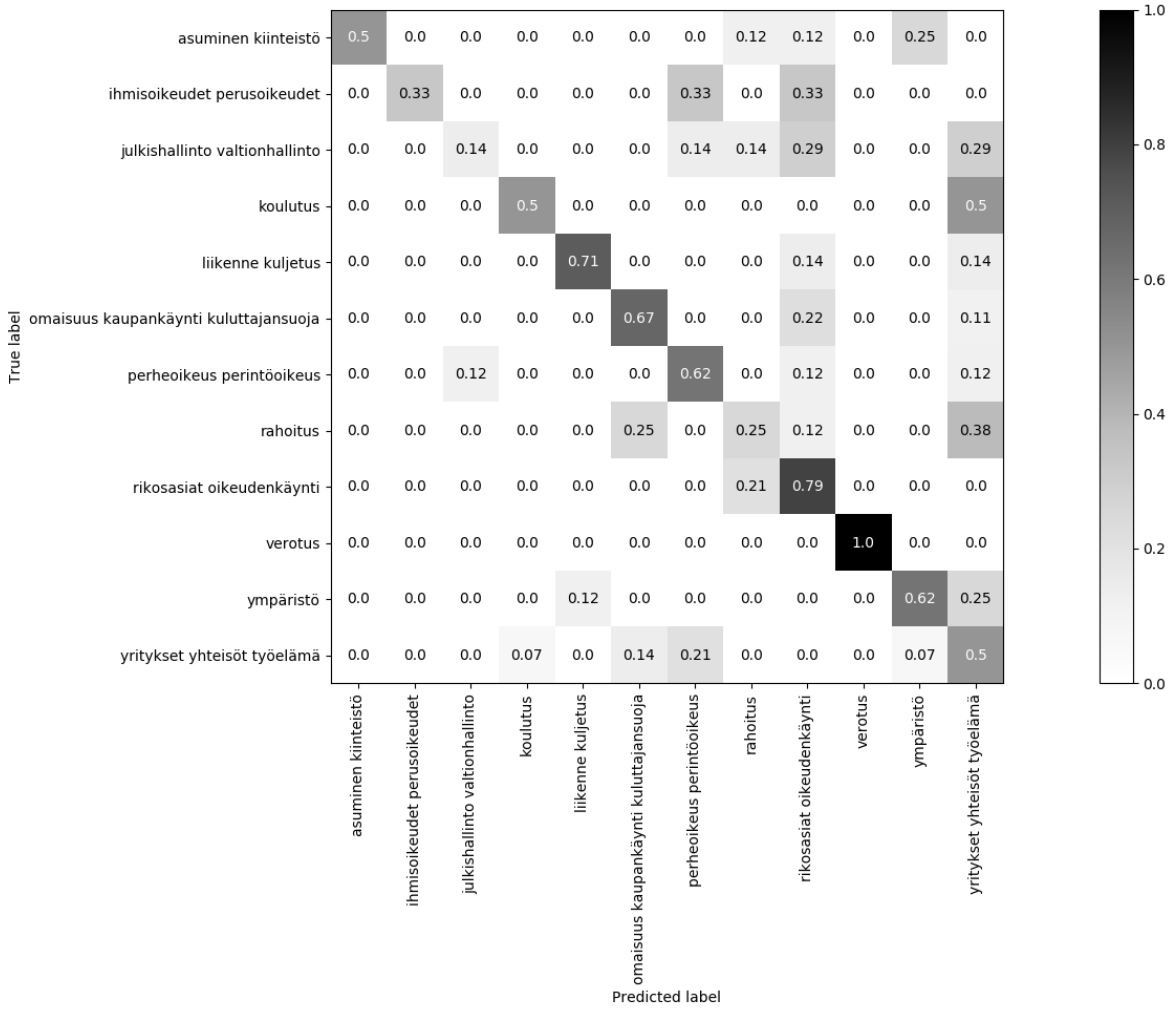


Figure 6.3: Confusion matrices: fastText by itself and using the Annif extractor (0.01) on Yle 2

[1]With the ‘none’ pooling strategy
[2]With the ‘none’ pooling strategy



fastText with SVD on Finlex

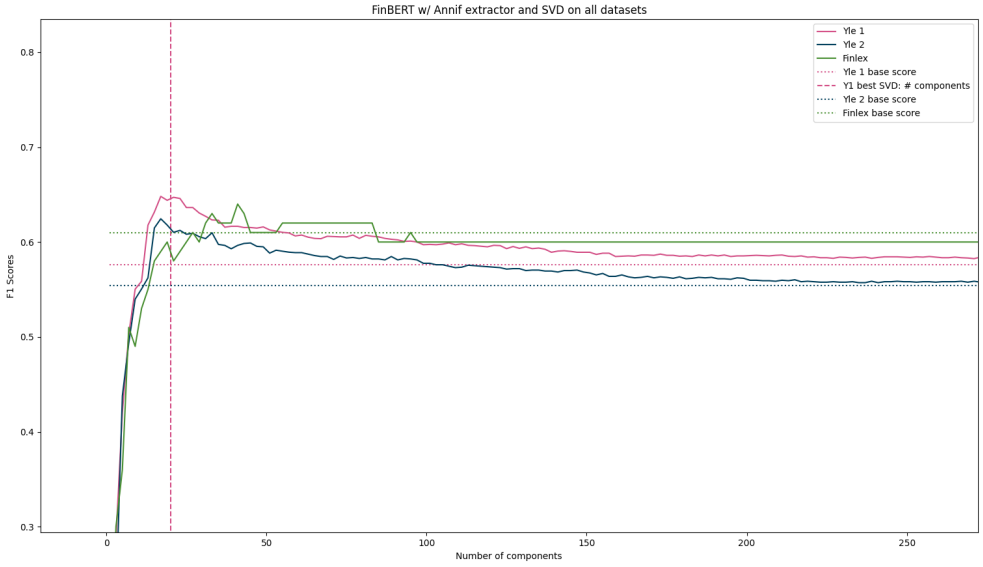


Figure 6.4: SVD components plot: FinBERT[1] w/ Annif and SVD

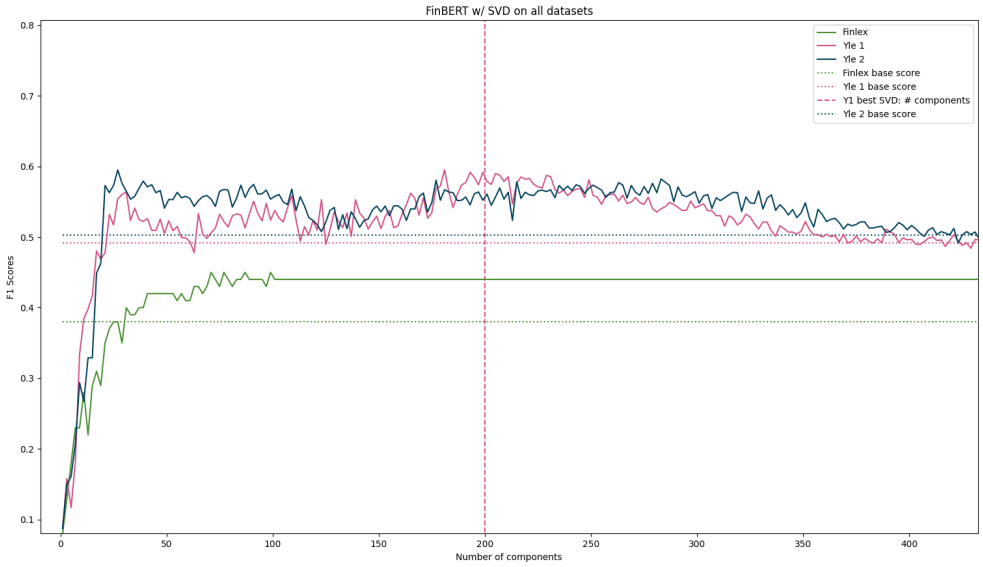


Figure 6.5: SVD components plot: FinBERT[2] w/ SVD (part)

Appendix B

These are the ids of the news articles that make up both Yle datasets.

Dataset: Yle 1

20-121178 20-121198 20-123679 20-123876 20-124067 20-127515 20-127849 20-131109 20-131536 20-131707 20-132992 20-133888 20-134736 20-134946 20-135406 20-137733 20-139003 20-139156 20-139745 20-140149 20-140931 20-141460 20-142165 20-142931 20-143562 20-144455 20-154226 20-154926 20-155114 20-155300 20-155913 20-156006 20-156881 20-158683 20-158943 20-164415 20-164849 20-165584 20-183742 20-184656 20-186044 20-187791 20-190842 20-191685 20-191752 20-192236 20-192873 20-193311 20-195029 20-196841 20-203489 20-211837 20-212187 20-212575 20-213224 20-251409 20-253388 20-253843 20-254895 20-255781 20-256293 20-257265 20-258496 20-258544 20-259364 20-261566 20-262958 20-263459 20-268030 20-268160 20-274596 20-275416 20-275840 20-276958 20-279424 20-279506 20-284661 20-289804 20-301448 20-303874 20-75608 20-77535 3-10001012 3-10001109 3-10001559 3-10003543 3-10017822 3-10017895 3-10019695 3-10023890 3-10028628 3-10029191 3-10032960 3-10034474 3-10035505 3-10038016 3-10038889 3-10039941 3-10048301 3-10048510 3-10049214 3-10050969 3-10051922 3-10052699 3-10053047 3-10054389 3-10055138 3-10058238 3-10058982 3-10059231 3-10059261 3-10060617 3-10060703 3-10061125 3-10061596 3-10062455 3-10063680 3-10065602 3-10078268 3-10082961 3-10092216 3-10093588 3-10094055 3-10095260 3-10096576 3-10099840 3-10114705 3-10120671 3-10128135 3-10128885 3-10130200 3-10134969 3-10135243 3-10142527 3-10149787 3-10152631 3-10154306 3-10160145 3-10164036 3-10164844 3-10166495 3-10167758 3-10168278 3-10169373 3-10171507 3-10178461 3-10181581 3-10187043 3-10187788 3-10195891 3-10197164 3-10203428 3-10207436 3-10209189 3-10216080 3-10217016 3-10219849 3-10237116 3-10245274 3-10247979 3-10248125 3-10249627 3-10251730 3-10254908 3-10260664 3-10265561 3-10265977 3-10275703 3-10291643 3-10304022 3-10312193 3-10317028 3-10319530 3-10320350 3-10329263 3-10330024 3-10336371

3-10339901 3-10341443 3-10342644 3-10342888 3-10343379 3-10346760 3-10351394 3-10351643
3-10355485 3-10357511 3-10362190 3-10368389 3-10369102 3-10373528 3-10375043 3-10380721
3-10386052 3-10387183 3-10388310 3-10396146 3-10398743 3-10398791 3-10405723 3-10405840
3-10410432 3-10415530 3-10419673 3-10421618 3-10424424 3-10431787 3-10433825 3-10438426
3-10441652 3-10442726 3-10466088 3-10474406 3-10474971 3-10475556 3-10476752 3-10478948
3-10479095 3-10479882 3-10480871 3-10498512 3-10499372 3-10501516 3-10501882 3-10504175
3-10505719 3-10514330 3-10517839 3-10522401 3-10526933 3-10529369 3-10531971 3-10533655
3-10533968 3-10534206 3-10541213 3-10541388 3-10541890 3-10541985 3-10542143 3-10542567
3-10542947 3-10552531 3-10554348 3-10560733 3-10565693 3-10566212 3-10575804 3-10575840
3-10580791 3-10582120 3-10586896 3-10589874 3-10591974 3-10603445 3-10611329 3-10613293
3-10621156 3-10625171 3-10626182 3-10627137 3-10632078 3-10633694 3-10637260 3-10646829
3-10652999 3-10656500 3-10656568 3-10661815 3-10666375 3-10671943 3-10689023 3-10695256
3-10697746 3-10698367 3-10702766 3-10703156 3-10709353 3-10713361 3-10718725 3-10722736
3-10723486 3-10725776 3-10729917 3-10734559 3-10744859 3-10777226 3-10785929 3-10793578
3-10795369 3-10797810 3-10805959 3-10806426 3-10806700 3-10808846 3-10818721 3-10823778
3-10830452 3-10830496 3-10838873 3-10850233 3-10850900 3-10851168 3-10853187 3-10857644
3-10872192 3-10873436 3-10874475 3-10879997 3-10881289 3-10888124 3-10890153 3-10894557
3-10894936 3-10894937 3-10895005 3-10898697 3-10899306 3-10907273 3-10908896 3-10915444
3-10918129 3-10919262 3-10919613 3-10919859 3-10920492 3-10924711 3-10929943 3-10931047
3-10934686 3-10934743 3-10935752 3-10940451 3-10946656 3-10946781 3-10947779 3-10958598
3-10972661 3-10977670 3-10979276 3-10979373 3-10981847 3-10983728 3-10984246 3-10986809
3-10988049 3-11003533 3-11006895 3-11010009 3-11011693 3-11014590 3-11019116 3-11022998
3-11026269 3-11028598 3-11029556 3-11033544 3-11035754 3-11043733 3-11047938 3-11050052
3-11054975 3-11072811 3-11075648 3-11076993 3-11099401 3-11119510 3-11142955 3-11148466
3-11149258 3-11154638 3-11172913 3-11178746 3-11179239 3-11179832 3-11186156 3-11198069
3-11198166 3-11198248 3-11198714 3-11204544 3-11206836 3-11234323 3-11235553 3-11237754
3-11253053 3-11254930 3-11261699 3-11272444 3-11276997 3-11281836 3-11286515 3-11293044
3-11294921 3-11310674 3-11310702 3-11311430 3-11313978 3-11315004 3-11316203 3-11317149
3-11317167 3-11317376 3-11317933 3-11323933 3-11324310 3-11325901 3-11325987 3-11328629
3-11329424 3-11332728 3-11334311 3-11336002 3-11337349 3-11338448 3-11339582 3-11340258
3-11340402 3-11340819 3-11340855 3-11345811 3-11346103 3-11353549 3-11353572 3-11359127
3-11366788 3-11371465 3-11371768 3-11374856 3-11375575 3-11375675 3-11378608 3-11381912

3-11382336 3-11387318 3-11404472 3-11408745 3-11412369 3-11417734 3-11421444 3-11425796
3-11445677 3-11445992 3-11446302 3-11446592 3-11447574 3-11450809 3-11455916 3-11456046
3-11466590 3-11472555 3-11474327 3-11477291 3-11477357 3-11477914 3-11480353 3-11499767
3-11503755 3-11506100 3-11514515 3-11515157 3-11534185 3-11538875 3-11541326 3-11544922
3-11552559 3-11556190 3-11557120 3-11559376 3-11562590 3-11568225 3-11569643 3-11576258
3-7641717 3-7662554 3-7700889 3-7702316 3-7703183 3-7705387 3-7709796 3-7712802 3-7714202
3-7714787 3-7716056 3-7716278 3-7716399 3-7716406 3-7716661 3-7716766 3-7716920 3-7716978
3-7717389 3-7717972 3-7719767 3-7720570 3-7720747 3-7720898 3-7721616 3-7721635 3-7721828
3-7721973 3-7722382 3-7722939 3-7723009 3-7723042 3-7723213 3-7723224 3-7723388 3-7723851
3-7723868 3-7723879 3-7724844 3-7724853 3-7724958 3-7725759 3-7725821 3-7725952 3-7725984
3-7726275 3-7726405 3-7726795 3-7727170 3-7727263 3-7727360 3-7727374 3-7727529 3-7727750
3-7727919 3-7727984 3-7728200 3-7728261 3-7728878 3-7729837 3-7730059 3-7730560 3-7731189
3-7731308 3-7731565 3-7731622 3-7731938 3-7731962 3-7732105 3-7733688 3-7733778 3-7734497
3-7734602 3-7734661 3-7734765 3-7735298 3-7735400 3-7735789 3-7735833 3-7735906 3-7736141
3-7736171 3-7736358 3-7736650 3-7736767 3-7736773 3-7736785 3-7737287 3-7737557 3-7738030
3-7738295 3-7738330 3-7738655 3-7739237 3-7739400 3-7739402 3-7739618 3-7739746 3-7740026
3-7740468 3-7740625 3-7740705 3-7740736 3-7741023 3-7741066 3-7741145 3-7741152 3-7741332
3-7741867 3-7741942 3-7742699 3-7743031 3-7743143 3-7743221 3-7743420 3-7743621 3-7743695
3-7743755 3-7744715 3-7744831 3-7745016 3-7745261 3-7745854 3-7746218 3-7746343 3-7746500
3-7746613 3-7747678 3-7748786 3-7748814 3-7748892 3-7749226 3-7749562 3-7750107 3-7750234
3-7751114 3-7751585 3-7751735 3-7751927 3-7752529 3-7752913 3-7753199 3-7753534 3-7753804
3-7755030 3-7755129 3-7755217 3-7755420 3-7755616 3-7755672 3-7755723 3-7755801 3-7756452
3-7757086 3-7759317 3-7759468 3-7760409 3-7760644 3-7761059 3-7761400 3-7761782 3-7761835
3-7761894 3-7761957 3-7761967 3-7761990 3-7762646 3-7762736 3-7762789 3-7762806 3-7762828
3-7763072 3-7763403 3-7763719 3-7763945 3-7764060 3-7765096 3-7765134 3-7765445 3-7765549
3-7765682 3-7765892 3-7766186 3-7766206 3-7766284 3-7766329 3-7767028 3-7767356 3-7767496
3-7767858 3-7768560 3-7768579 3-7768713 3-7768725 3-7768737 3-7769636 3-7769764 3-7769828
3-7770221 3-7770249 3-7770273 3-7770942 3-7770951 3-7771164 3-7772693 3-7773132 3-7773257
3-7773974 3-7774199 3-7774479 3-7774613 3-7774693 3-7774792 3-7775112 3-7775130 3-7775131
3-7775676 3-7776150 3-7776997 3-7777709 3-7777849 3-7777988 3-7778048 3-7778060 3-7778331
3-7778576 3-7778596 3-7778712 3-7778818 3-7779386 3-7779623 3-7780552 3-7780816 3-7781492
3-7781640 3-7781711 3-7781712 3-7781918 3-7782262 3-7783229 3-7783389 3-7783401 3-7783448

3-7783839 3-7783913 3-7784411 3-7784542 3-7784667 3-7785175 3-7785548 3-7786383 3-7786405
3-7786681 3-7787112 3-7787143 3-7787181 3-7787267 3-7787294 3-7787314 3-7788103 3-7789074
3-7789700 3-7789798 3-7790091 3-7790381 3-7790612 3-7790724 3-7791128 3-7791326 3-7791773
3-7792095 3-7792159 3-7792196 3-7792716 3-7793675 3-7793795 3-7793907 3-7793918 3-7793965
3-7794145 3-7794234 3-7794327 3-7795410 3-7796758 3-7797262 3-7797292 3-7797501 3-7797586
3-7798349 3-7798563 3-7798689 3-7798899 3-7799046 3-7799246 3-7799555 3-7800031 3-7800081
3-7800220 3-7800324 3-7800497 3-7800575 3-7800873 3-7800911 3-7801240 3-7801248 3-7802150
3-7802671 3-7802678 3-7802912 3-7802947 3-7803151 3-7803553 3-7804034 3-7804381 3-7804746
3-7805022 3-7805426 3-7806021 3-7806667 3-7807175 3-7807205 3-7807332 3-7807842 3-7808152
3-7808162 3-7809004 3-7809641 3-7809841 3-7809987 3-7810573 3-7810575 3-7810752 3-7811559
3-7811795 3-7812513 3-7813049 3-7813053 3-7813224 3-7813244 3-7813401 3-7813561 3-7813618
3-7813633 3-7813803 3-7814486 3-7814879 3-7814946 3-7815369 3-7815900 3-7816699 3-7816716
3-7816760 3-7816769 3-7816795 3-7817490 3-7818052 3-7818103 3-7818123 3-7819181 3-7819185
3-7819656 3-7819722 3-7820070 3-7822857 3-7823313 3-7823321 3-7823563 3-7823566 3-7824191
3-7824488 3-7824950 3-7825156 3-7825242 3-7825314 3-7825368 3-7825543 3-7825598 3-7825653
3-7826015 3-7826097 3-7826295 3-7826330 3-7826338 3-7826480 3-7827184 3-7827297 3-7827740
3-7827893 3-7828007 3-7828165 3-7828222 3-7828392 3-7828424 3-7828428 3-7828573 3-7828645
3-7828769 3-7828771 3-7828911 3-7829130 3-7829813 3-7830493 3-7830560 3-7830631 3-7830705
3-7830707 3-7830825 3-7830833 3-7830884 3-7831278 3-7831310 3-7831759 3-7831777 3-7832049
3-7832095 3-7832127 3-7832259 3-7833045 3-7833084 3-7833307 3-7833524 3-7834038 3-7834284
3-7834625 3-7834736 3-7835580 3-7836664 3-7838310 3-7838447 3-7838867 3-7838964 3-7839291
3-7839328 3-7839544 3-7839696 3-7839716 3-7840833 3-7841005 3-7841016 3-7841854 3-7841920
3-7841977 3-7842116 3-7842317 3-7842465 3-7843165 3-7843933 3-7844164 3-7844167 3-7844339
3-7844365 3-7844424 3-7844435 3-7844556 3-7845018 3-7845284 3-7845381 3-7845455 3-7846754
3-7846835 3-7846868 3-7847026 3-7847864 3-7848006 3-7848121 3-7848530 3-7849339 3-7849586
3-7849731 3-7849881 3-7850152 3-7850363 3-7850469 3-7850600 3-7850902 3-7851298 3-7851510
3-7851851 3-7853358 3-7853844 3-7854037 3-7854260 3-7854883 3-7855226 3-7855243 3-7855631
3-7855926 3-7855967 3-7856837 3-7856928 3-7856956 3-7857174 3-7857277 3-7857341 3-7857500
3-7857554 3-7857657 3-7857785 3-7858130 3-7858182 3-7858351 3-7858578 3-7858958 3-7859182
3-7859750 3-7860316 3-7860365 3-7860394 3-7860741 3-7860852 3-7861714 3-7861723 3-7862367
3-7862506 3-7862828 3-7862858 3-7862973 3-7863180 3-7863185 3-7863583 3-7863746 3-7863821
3-7864650 3-7864880 3-7864972 3-7864996 3-7865070 3-7865249 3-7865332 3-7865420 3-7865493

3-7865715 3-7866061 3-7866767 3-7866861 3-7867379 3-7867639 3-7867661 3-7867690 3-7867962
3-7868672 3-7868739 3-7869541 3-7869553 3-7869602 3-7869758 3-7870107 3-7870217 3-7870365
3-7870485 3-7870643 3-7870683 3-7870769 3-7870831 3-7870839 3-7870842 3-7871133 3-7871779
3-7871889 3-7872507 3-7872672 3-7872943 3-7872982 3-7873283 3-7873319 3-7873545 3-7873561
3-7873895 3-7874385 3-7874708 3-7874909 3-7874995 3-7875193 3-7875246 3-7875712 3-7876044
3-7876061 3-7876094 3-7876239 3-7876276 3-7877127 3-7877306 3-7877961 3-7877966 3-7878045
3-7879140 3-7879155 3-7879216 3-7879266 3-7879425 3-7879434 3-7880651 3-7880884 3-7880985
3-7880991 3-7880999 3-7881283 3-7881383 3-7881580 3-7882759 3-7883147 3-7883309 3-7883324
3-7884204 3-7884759 3-7884814 3-7885330 3-7885415 3-7886668 3-7886916 3-7886927 3-7887330
3-7887559 3-7887578 3-7887869 3-7888231 3-7888251 3-7888343 3-7888386 3-7888635 3-7888690
3-7889053 3-7889196 3-7890734 3-7890736 3-7890837 3-7891284 3-7891415 3-7891827 3-7893222
3-7893231 3-7893454 3-7893590 3-7893608 3-7894668 3-7894724 3-7898084 3-7898157 3-7899448
3-7899565 3-7900076 3-7900120 3-7900279 3-7900282 3-7900330 3-7900515 3-7901079 3-7901092
3-7901098 3-7901337 3-7901387 3-7901412 3-7901487 3-7902125 3-7902237 3-7902501 3-7902711
3-7903074 3-7903663 3-7904911 3-7905033 3-7906322 3-7907298 3-7907727 3-7908771 3-7909977
3-7910558 3-7910695 3-7910755 3-7911175 3-7911482 3-7911603 3-7911794 3-7911928 3-7912198
3-7912424 3-7912664 3-7913011 3-7914052 3-7914191 3-7915218 3-7915335 3-7916084 3-7916389
3-7916434 3-7917124 3-7917339 3-7917388 3-7917979 3-7918531 3-7918621 3-7919630 3-7919639
3-7920174 3-7920243 3-7920301 3-7920605 3-7921409 3-7921773 3-7921930 3-7922689 3-7923454
3-7924058 3-7924595 3-7924621 3-7924917 3-7925426 3-7925705 3-7925747 3-7925754 3-7926122
3-7926847 3-7926999 3-7927033 3-7927137 3-7927709 3-7927719 3-7927915 3-7927919 3-7927970
3-7928134 3-7928192 3-7928276 3-7928376 3-7928953 3-7928960 3-7929097 3-7929323 3-7929383
3-7929783 3-7929884 3-7930492 3-7930568 3-7930985 3-7932354 3-7932396 3-7932846 3-7933310
3-7933673 3-7934008 3-7934536 3-7935003 3-7935269 3-7935665 3-7935711 3-7936183 3-7936851
3-7937134 3-7937696 3-7941443 3-7941854 3-7942063 3-7942889 3-7942983 3-7943335 3-7943622
3-7944010 3-7944057 3-7944203 3-7945757 3-7946018 3-7946462 3-7946833 3-7947120 3-7948511
3-7949055 3-7949321 3-7949416 3-7949627 3-7950747 3-7951288 3-7951330 3-7951391 3-7952542
3-7952608 3-7952771 3-7953030 3-7953143 3-7953383 3-7953442 3-7954268 3-7954755 3-7955339
3-7955726 3-7956151 3-7956944 3-7958155 3-7958366 3-7958549 3-7958821 3-7959242 3-7960899
3-7961395 3-7962176 3-7962192 3-7963252 3-7963563 3-7964293 3-7964319 3-7964688 3-7965272
3-7969321 3-7969445 3-7969459 3-7969473 3-7970190 3-7970250 3-7970480 3-7970767 3-7971485
3-7972695 3-7973963 3-7974617 3-7974733 3-7975178 3-7975303 3-7975435 3-7975613 3-7975711

3-7975811 3-7977426 3-7977686 3-7977879 3-7977951 3-7977965 3-7978171 3-7978290 3-7978495
3-7978586 3-7979042 3-7980105 3-7981853 3-7984935 3-7985585 3-7985841 3-7985973 3-7986240
3-7987151 3-7987540 3-7987752 3-7987800 3-7988595 3-7988741 3-7988851 3-7989060 3-7990023
3-7990607 3-7990621 3-7990999 3-7991452 3-7992018 3-7992262 3-7993093 3-7993356 3-7994361
3-7994910 3-7995427 3-7995648 3-7995899 3-7997664 3-7997913 3-7998856 3-7999662 3-7999712
3-8000375 3-8000924 3-8001347 3-8001412 3-8001454 3-8001484 3-8002069 3-8002415 3-8002862
3-8003200 3-8003211 3-8003689 3-8004251 3-8004870 3-8004888 3-8004945 3-8005473 3-8006321
3-8006900 3-8007034 3-8008016 3-8008546 3-8008819 3-8009082 3-8009470 3-8009570 3-8009727
3-8010743 3-8011370 3-8012946 3-8013590 3-8014739 3-8015394 3-8015962 3-8016838 3-8017786
3-8017841 3-8018686 3-8018810 3-8019086 3-8019467 3-8020068 3-8021554 3-8022334 3-8023759
3-8024134 3-8024829 3-8025608 3-8025793 3-8026055 3-8026282 3-8027523 3-8028147 3-8028586
3-8029645 3-8029670 3-8033237 3-8033507 3-8033518 3-8033698 3-8034654 3-8034681 3-8035330
3-8036367 3-8037843 3-8038126 3-8039255 3-8039943 3-8040957 3-8041424 3-8041760 3-8042427
3-8042447 3-8042575 3-8042708 3-8043783 3-8044214 3-8044672 3-8044855 3-8045076 3-8045588
3-8045612 3-8045922 3-8046347 3-8046694 3-8048407 3-8049218 3-8049886 3-8050092 3-8050256
3-8055187 3-8055629 3-8055999 3-8056529 3-8057155 3-8057390 3-8057999 3-8058855 3-8061928
3-8062995 3-8063071 3-8063454 3-8063967 3-8064019 3-8064519 3-8065685 3-8066028 3-8067127
3-8067249 3-8067307 3-8067658 3-8069850 3-8070009 3-8070012 3-8070057 3-8070083 3-8070150
3-8070322 3-8071000 3-8071329 3-8072728 3-8073288 3-8074786 3-8074817 3-8074856 3-8075423
3-8075507 3-8076406 3-8078192 3-8078450 3-8078741 3-8078859 3-8079593 3-8079710 3-8079895
3-8080096 3-8082110 3-8082719 3-8082825 3-8083193 3-8083264 3-8083311 3-8084546 3-8085985
3-8086199 3-8086212 3-8086242 3-8086256 3-8086886 3-8087052 3-8087324 3-8089367 3-8091727
3-8092398 3-8093859 3-8093891 3-8094225 3-8094570 3-8095048 3-8095398 3-8095578 3-8095836
3-8095886 3-8096701 3-8097487 3-8099836 3-8100162 3-8100572 3-8101212 3-8102882 3-8102925
3-8103525 3-8104141 3-8104400 3-8104915 3-8105865 3-8106206 3-8106251 3-8106709 3-8107332
3-8108739 3-8109371 3-8109880 3-8110382 3-8110539 3-8111084 3-8114461 3-8114903 3-8116101
3-8117663 3-8118276 3-8118423 3-8119127 3-8119642 3-8119944 3-8120909 3-8120929 3-8120955
3-8121476 3-8121495 3-8121993 3-8122023 3-8122713 3-8123069 3-8123335 3-8123480 3-8124816
3-8125019 3-8127677 3-8128670 3-8130529 3-8131146 3-8131322 3-8132092 3-8132139 3-8132243
3-8132260 3-8132468 3-8132519 3-8135621 3-8136323 3-8137792 3-8138012 3-8138351 3-8138510
3-8138540 3-8138978 3-8139050 3-8139643 3-8141897 3-8142395 3-8143543 3-8144344 3-8145384
3-8145937 3-8149151 3-8150996 3-8153025 3-8153687 3-8154013 3-8154654 3-8155087 3-8155211

3-8156735 3-8156935 3-8157012 3-8157476 3-8157682 3-8157789 3-8157947 3-8158196 3-8158383
3-8160228 3-8160846 3-8162101 3-8162570 3-8162600 3-8162902 3-8163043 3-8163170 3-8163449
3-8163691 3-8163824 3-8164837 3-8165004 3-8165014 3-8166608 3-8167196 3-8167296 3-8167514
3-8167667 3-8168219 3-8168413 3-8169603 3-8170606 3-8170735 3-8170908 3-8170934 3-8171899
3-8172340 3-8172698 3-8172716 3-8173314 3-8173452 3-8174818 3-8175441 3-8175623 3-8175668
3-8175678 3-8176087 3-8176146 3-8176538 3-8178311 3-8178799 3-8178950 3-8179170 3-8179742
3-8180063 3-8183118 3-8183145 3-8184390 3-8184815 3-8186381 3-8186391 3-8187901 3-8188282
3-8189290 3-8189764 3-8191578 3-8191608 3-8191917 3-8192659 3-8193647 3-8194875 3-8194944
3-8195595 3-8195737 3-8196759 3-8198424 3-8199850 3-8199929 3-8200430 3-8200597 3-8200640
3-8200663 3-8201316 3-8201860 3-8201881 3-8202064 3-8202920 3-8203336 3-8204323 3-8207244
3-8207474 3-8207554 3-8207972 3-8208032 3-8208101 3-8208369 3-8209034 3-8209295 3-8209705
3-8210091 3-8210462 3-8210753 3-8210994 3-8211298 3-8212498 3-8213151 3-8213355 3-8214797
3-8215330 3-8216045 3-8216089 3-8216506 3-8217045 3-8217167 3-8217201 3-8217299 3-8217627
3-8217661 3-8218353 3-8218420 3-8219236 3-8220689 3-8221276 3-8222337 3-8223311 3-8223666
3-8224653 3-8225431 3-8227384 3-8227721 3-8227989 3-8228139 3-8228598 3-8229571 3-8231329
3-8231381 3-8232089 3-8232773 3-8232856 3-8233041 3-8233335 3-8233420 3-8233427 3-8233590
3-8234401 3-8234681 3-8234762 3-8234887 3-8235030 3-8235921 3-8235965 3-8235994 3-8236187
3-8236351 3-8237919 3-8238060 3-8239053 3-8239536 3-8241788 3-8241867 3-8244349 3-8244865
3-8246412 3-8247448 3-8248256 3-8249796 3-8249899 3-8249961 3-8250362 3-8251599 3-8252266
3-8252483 3-8252920 3-8254036 3-8254148 3-8255143 3-8255388 3-8255950 3-8257500 3-8258656
3-8258657 3-8258683 3-8259290 3-8259409 3-8261529 3-8261839 3-8262811 3-8263168 3-8265562
3-8266472 3-8266556 3-8266911 3-8267870 3-8268074 3-8268366 3-8268608 3-8269892 3-8270027
3-8270184 3-8271056 3-8271200 3-8271253 3-8271962 3-8272766 3-8272861 3-8272957 3-8273443
3-8274505 3-8274606 3-8276316 3-8276347 3-8276944 3-8279320 3-8279417 3-8279771 3-8280079
3-8280123 3-8280321 3-8281486 3-8281967 3-8282629 3-8282736 3-8283000 3-8284535 3-8285275
3-8285589 3-8286614 3-8287092 3-8288707 3-8289629 3-8291901 3-8294062 3-8294110 3-8295154
3-8295183 3-8295205 3-8295323 3-8296006 3-8296152 3-8296251 3-8296452 3-8297723 3-8297987
3-8298512 3-8299717 3-8299984 3-8300449 3-8300789 3-8301190 3-8303281 3-8303326 3-8303360
3-8303662 3-8303935 3-8304653 3-8306531 3-8307505 3-8308806 3-8309188 3-8309583 3-8309627
3-8311415 3-8311806 3-8312095 3-8312347 3-8312382 3-8312986 3-8313137 3-8313691 3-8314251
3-8314345 3-8315301 3-8315574 3-8315814 3-8317229 3-8317672 3-8319324 3-8319849 3-8320199
3-8320619 3-8320806 3-8321885 3-8322757 3-8323319 3-8324910 3-8325294 3-8325845 3-8326432

3-8328379 3-8328863 3-8329134 3-8329497 3-8329608 3-8329728 3-8331135 3-8331192 3-8331718
3-8331722 3-8331815 3-8331865 3-8332518 3-8334472 3-8336147 3-8336596 3-8337084 3-8337648
3-8337798 3-8338591 3-8339149 3-8340178 3-8340452 3-8341305 3-8341849 3-8342471 3-8342927
3-8343212 3-8343669 3-8344300 3-8344802 3-8344813 3-8345221 3-8346397 3-8346539 3-8347646
3-8347703 3-8348042 3-8348346 3-8348474 3-8348674 3-8349192 3-8349418 3-8350484 3-8350689
3-8351121 3-8351806 3-8352927 3-8354472 3-8354628 3-8354707 3-8354870 3-8355076 3-8357797
3-8358367 3-8358783 3-8359380 3-8359575 3-8360117 3-8360183 3-8360620 3-8361252 3-8361718
3-8362884 3-8363501 3-8363764 3-8364593 3-8364737 3-8365251 3-8365419 3-8365739 3-8366107
3-8367926 3-8368300 3-8368754 3-8369156 3-8369428 3-8370166 3-8371087 3-8371117 3-8373200
3-8373708 3-8374049 3-8374723 3-8375589 3-8376122 3-8376309 3-8376410 3-8379821 3-8380701
3-8380763 3-8381542 3-8381768 3-8382668 3-8384444 3-8384896 3-8384930 3-8385551 3-8387844
3-8388282 3-8389908 3-8390427 3-8390478 3-8391029 3-8391084 3-8391833 3-8391889 3-8392412
3-8392485 3-8392979 3-8393126 3-8393197 3-8393235 3-8395254 3-8396095 3-8396252 3-8396452
3-8396482 3-8396696 3-8396840 3-8398201 3-8399693 3-8400861 3-8401356 3-8402461 3-8402714
3-8402749 3-8402795 3-8404452 3-8407507 3-8407701 3-8408538 3-8409315 3-8410286 3-8410317
3-8410796 3-8411325 3-8411759 3-8412914 3-8413347 3-8413556 3-8413660 3-8414347 3-8414359
3-8414663 3-8414699 3-8415475 3-8415528 3-8415639 3-8415740 3-8416234 3-8416241 3-8416502
3-8416879 3-8417582 3-8417760 3-8418600 3-8418778 3-8419256 3-8419674 3-8420875 3-8421412
3-8421651 3-8423372 3-8424251 3-8424382 3-8425351 3-8425381 3-8427719 3-8427817 3-8427825
3-8427830 3-8428473 3-8428956 3-8428976 3-8428988 3-8429135 3-8429957 3-8430210 3-8431084
3-8431228 3-8431669 3-8431690 3-8431752 3-8432865 3-8432924 3-8433092 3-8433324 3-8433450
3-8434096 3-8434529 3-8435032 3-8435738 3-8436114 3-8436305 3-8437081 3-8437402 3-8438413
3-8440321 3-8440414 3-8440478 3-8441193 3-8441463 3-8441905 3-8442335 3-8443798 3-8444318
3-8444692 3-8445047 3-8445079 3-8445783 3-8448139 3-8448310 3-8448851 3-8450338 3-8450790
3-8450855 3-8451639 3-8451835 3-8452874 3-8453549 3-8455091 3-8456057 3-8459607 3-8460684
3-8460707 3-8461856 3-8463368 3-8463448 3-8464049 3-8464801 3-8466423 3-8466590 3-8466885
3-8467047 3-8467078 3-8467190 3-8468126 3-8468573 3-8468957 3-8469519 3-8469830 3-8471494
3-8471793 3-8473134 3-8473466 3-8474382 3-8474466 3-8479504 3-8479862 3-8480725 3-8480926
3-8481282 3-8482672 3-8482690 3-8483975 3-8484215 3-8484223 3-8485971 3-8487297 3-8487324
3-8487441 3-8487843 3-8487944 3-8488308 3-8490937 3-8491680 3-8492053 3-8492181 3-8494132
3-8494783 3-8496633 3-8497064 3-8498170 3-8498589 3-8499324 3-8500153 3-8500966 3-8501345
3-8501734 3-8502520 3-8503015 3-8506792 3-8507166 3-8510818 3-8513027 3-8513418 3-8514459

3-8515369 3-8515600 3-8517891 3-8519059 3-8526083 3-8526887 3-8527956 3-8528740 3-8529475
3-8533667 3-8534497 3-8534739 3-8534755 3-8535855 3-8536865 3-8537245 3-8537946 3-8538302
3-8541767 3-8542892 3-8543035 3-8544079 3-8544459 3-8544650 3-8545762 3-8547568 3-8547652
3-8550814 3-8552677 3-8553163 3-8553757 3-8554714 3-8554739 3-8554906 3-8555855 3-8555887
3-8556009 3-8557235 3-8560045 3-8563473 3-8563493 3-8564921 3-8565199 3-8565379 3-8568380
3-8568935 3-8569918 3-8571577 3-8577263 3-8578589 3-8579388 3-8585784 3-8587594 3-8587650
3-8588216 3-8588274 3-8590443 3-8590606 3-8591889 3-8593652 3-8593655 3-8594682 3-8595115
3-8597007 3-8598181 3-8598657 3-8599615 3-8602893 3-8604565 3-8608567 3-8609498 3-8613204
3-8615265 3-8616210 3-8616480 3-8616727 3-8617905 3-8617932 3-8622113 3-8622379 3-8624040
3-8624124 3-8624457 3-8626775 3-8628299 3-8628983 3-8634211 3-8640116 3-8640801 3-8642849
3-8643574 3-8645924 3-8648353 3-8652259 3-8659054 3-8663011 3-8663587 3-8665927 3-8666092
3-8668479 3-8669446 3-8669447 3-8672452 3-8673600 3-8674102 3-8677332 3-8681382 3-8682154
3-8682483 3-8683061 3-8686172 3-8686519 3-8687275 3-8689895 3-8693246 3-8696767 3-8697271
3-8698250 3-8699341 3-8702324 3-8703837 3-8705879 3-8706821 3-8709814 3-8712113 3-8712983
3-8714499 3-8714588 3-8716172 3-8716634 3-8717242 3-8717390 3-8719975 3-8720012 3-8724474
3-8725751 3-8726843 3-8727358 3-8731263 3-8732416 3-8732429 3-8733108 3-8734952 3-8736989
3-8739267 3-8741000 3-8742297 3-8742589 3-8749415 3-8752855 3-8757069 3-8758763 3-8759678
3-8759701 3-8761111 3-8762809 3-8763014 3-8763286 3-8763525 3-8765432 3-8768840 3-8769992
3-8772226 3-8773315 3-8774131 3-8775701 3-8778004 3-8778232 3-8782153 3-8783702 3-8784530
3-8784981 3-8785183 3-8786727 3-8788986 3-8789786 3-8790555 3-8792174 3-8792281 3-8793508
3-8795012 3-8802856 3-8803484 3-8805216 3-8805259 3-8806224 3-8807285 3-8807923 3-8808589
3-8811284 3-8811820 3-8813722 3-8816408 3-8816583 3-8817494 3-8822054 3-8822962 3-8824157
3-8824603 3-8825332 3-8827532 3-8829477 3-8830358 3-8830647 3-8830987 3-8833159 3-8834235
3-8834348 3-8834753 3-8837070 3-8839712 3-8840894 3-8841534 3-8842494 3-8842735 3-8843591
3-8844634 3-8844797 3-8844828 3-8847375 3-8850228 3-8850649 3-8850873 3-8851498 3-8851963
3-8852401 3-8854935 3-8858281 3-8858790 3-8861439 3-8861592 3-8862097 3-8862368 3-8862427
3-8866270 3-8867474 3-8868185 3-8869979 3-8870639 3-8871555 3-8873671 3-8876601 3-8877025
3-8877615 3-8881334 3-8885296 3-8887751 3-8888311 3-8890006 3-8890245 3-8891104 3-8893105
3-8893397 3-8894922 3-8895551 3-8898069 3-8898203 3-8900251 3-8900857 3-8903789 3-8908266
3-8911089 3-8911326 3-8911541 3-8914029 3-8916011 3-8917038 3-8917047 3-8920819 3-8921015
3-8922526 3-8924838 3-8924888 3-8926713 3-8927876 3-8928685 3-8928989 3-8932805 3-8936716
3-8938907 3-8939295 3-8940190 3-8940578 3-8941158 3-8941432 3-8942068 3-8942612 3-8942671

3-8948283 3-8948968 3-8950197 3-8951744 3-8953789 3-8954947 3-8955056 3-8955219 3-8955478
3-8955536 3-8956068 3-8956503 3-8956544 3-8958133 3-8958639 3-8958691 3-8959296 3-8961176
3-8962161 3-8964462 3-8964474 3-8965884 3-8966044 3-8968217 3-8968320 3-8973534 3-8975174
3-8975460 3-8975539 3-8975651 3-8975732 3-8977109 3-8977539 3-8983483 3-8983899 3-8984533
3-8984823 3-8988111 3-8989189 3-8990909 3-8995432 3-8996393 3-8999087 3-9000452 3-9001129
3-9001415 3-9001425 3-9002658 3-9004354 3-9004828 3-9005248 3-9005419 3-9006039 3-9006719
3-9008351 3-9012522 3-9013533 3-9015681 3-9018390 3-9019084 3-9019729 3-9020349 3-9020572
3-9026011 3-9026608 3-9036207 3-9037374 3-9041568 3-9045163 3-9050906 3-9051400 3-9052067
3-9052555 3-9053987 3-9054394 3-9055617 3-9056112 3-9056412 3-9059703 3-9060410 3-9061050
3-9061380 3-9062475 3-9062971 3-9063005 3-9063809 3-9064013 3-9064112 3-9064329 3-9064419
3-9064542 3-9064684 3-9064737 3-9064883 3-9064979 3-9065060 3-9065169 3-9065450 3-9065471
3-9066096 3-9066422 3-9068654 3-9069924 3-9070643 3-9070664 3-9080019 3-9080506 3-9081153
3-9081200 3-9082776 3-9084739 3-9084815 3-9085716 3-9086502 3-9087768 3-9088662 3-9089172
3-9089643 3-9091500 3-9095004 3-9095132 3-9095603 3-9097675 3-9097990 3-9098539 3-9099000
3-9099962 3-9104710 3-9106239 3-9110387 3-9110584 3-9111036 3-9111936 3-9114164 3-9115195
3-9116139 3-9117587 3-9117981 3-9119787 3-9120233 3-9120698 3-9122599 3-9124072 3-9124551
3-9124646 3-9124683 3-9124989 3-9125133 3-9128777 3-9130037 3-9130321 3-9130996 3-9133198
3-9133304 3-9133373 3-9133384 3-9133389 3-9133424 3-9133649 3-9133654 3-9133819 3-9133830
3-9133863 3-9134019 3-9134174 3-9134260 3-9134279 3-9134470 3-9134550 3-9134571 3-9134748
3-9134993 3-9135207 3-9135319 3-9136032 3-9136325 3-9136532 3-9136666 3-9136801 3-9136941
3-9137057 3-9137107 3-9137201 3-9137334 3-9137848 3-9138163 3-9139279 3-9144652 3-9146697
3-9147292 3-9147564 3-9147598 3-9147898 3-9152690 3-9153431 3-9154961 3-9160276 3-9160692
3-9161378 3-9162291 3-9164471 3-9164548 3-9164631 3-9165992 3-9166915 3-9168877 3-9169402
3-9170133 3-9177509 3-9177777 3-9177862 3-9178911 3-9180206 3-9180355 3-9180784 3-9180858
3-9181460 3-9182261 3-9183097 3-9183279 3-9184857 3-9185269 3-9185428 3-9186015 3-9186271
3-9186843 3-9187077 3-9187146 3-9187296 3-9187743 3-9188234 3-9188323 3-9188334 3-9188409
3-9189000 3-9189683 3-9190002 3-9190520 3-9190935 3-9191260 3-9191265 3-9191320 3-9191411
3-9191472 3-9191506 3-9191720 3-9191736 3-9191749 3-9191978 3-9192695 3-9193175 3-9193819
3-9193840 3-9193901 3-9194023 3-9194060 3-9194161 3-9194206 3-9194462 3-9194464 3-9194582
3-9194715 3-9194723 3-9194770 3-9194799 3-9194807 3-9195128 3-9195171 3-9195211 3-9195256
3-9196050 3-9196303 3-9196350 3-9196457 3-9196493 3-9196594 3-9196639 3-9196688 3-9196714
3-9197061 3-9197184 3-9197206 3-9197688 3-9197917 3-9197943 3-9198344 3-9198951 3-9199038

3-9199066 3-9199206 3-9199263 3-9199317 3-9199584 3-9199652 3-9199659 3-9199682 3-9199872
3-9199933 3-9200086 3-9200168 3-9200195 3-9200197 3-9200414 3-9200442 3-9200595 3-9200667
3-9200937 3-9200991 3-9201142 3-9201441 3-9201661 3-9201663 3-9201754 3-9202000 3-9202093
3-9202446 3-9202462 3-9202743 3-9203021 3-9203379 3-9203955 3-9203966 3-9204198 3-9204530
3-9204564 3-9205843 3-9205858 3-9205905 3-9205917 3-9205959 3-9206047 3-9207832 3-9208177
3-9208806 3-9209393 3-9210363 3-9210535 3-9211622 3-9211788 3-9212438 3-9213888 3-9214682
3-9215760 3-9215914 3-9216066 3-9219814 3-9221080 3-9221516 3-9224605 3-9224692 3-9224897
3-9226236 3-9227136 3-9227808 3-9229142 3-9229714 3-9229804 3-9229859 3-9231440 3-9234454
3-9234632 3-9235038 3-9237532 3-9237736 3-9240278 3-9241255 3-9242255 3-9243366 3-9243474
3-9243966 3-9246653 3-9247664 3-9248496 3-9249222 3-9249271 3-9249997 3-9250307 3-9250522
3-9250887 3-9251208 3-9252438 3-9253929 3-9255268 3-9255298 3-9255984 3-9256011 3-9256154
3-9256977 3-9257528 3-9257707 3-9258633 3-9260583 3-9264566 3-9264827 3-9264850 3-9265705
3-9268708 3-9270804 3-9272947 3-9274884 3-9275917 3-9276402 3-9277535 3-9278150 3-9278541
3-9278584 3-9279009 3-9279545 3-9279572 3-9281571 3-9281875 3-9282787 3-9282890 3-9283618
3-9284234 3-9285678 3-9286689 3-9287880 3-9288463 3-9288795 3-9292240 3-9292878 3-9293227
3-9294921 3-9295016 3-9296451 3-9299191 3-9302978 3-9304452 3-9307620 3-9310329 3-9310641
3-9311850 3-9313553 3-9314034 3-9314128 3-9315449 3-9315512 3-9315525 3-9315750 3-9318320
3-9319219 3-9321958 3-9323252 3-9324979 3-9325332 3-9328676 3-9330378 3-9336004 3-9340179
3-9343601 3-9343971 3-9345238 3-9347153 3-9349715 3-9351116 3-9351792 3-9352187 3-9356654
3-9362355 3-9364171 3-9366454 3-9377454 3-9377827 3-9378905 3-9383999 3-9385814 3-9386744
3-9386913 3-9389205 3-9391304 3-9393567 3-9397870 3-9399665 3-9400379 3-9400858 3-9401883
3-9408106 3-9409717 3-9412468 3-9412782 3-9412920 3-9413312 3-9414006 3-9414710 3-9415114
3-9416451 3-9417384 3-9418364 3-9419782 3-9419948 3-9419986 3-9420451 3-9421310 3-9423558
3-9424686 3-9425624 3-9427502 3-9427524 3-9427898 3-9430273 3-9433012 3-9435365 3-9435419
3-9437295 3-9442426 3-9442530 3-9442771 3-9443354 3-9444039 3-9446127 3-9446613 3-9447593
3-9449101 3-9451912 3-9452721 3-9455325 3-9456573 3-9457227 3-9457305 3-9457718 3-9460014
3-9460943 3-9461260 3-9464188 3-9465891 3-9467936 3-9468001 3-9468778 3-9469892 3-9471231
3-9471276 3-9475563 3-9478442 3-9482849 3-9493279 3-9494281 3-9497101 3-9498328 3-9500734
3-9503056 3-9505337 3-9511997 3-9514444 3-9516507 3-9520198 3-9521556 3-9523977 3-9524220
3-9524420 3-9524545 3-9526646 3-9533265 3-9539555 3-9540630 3-9543402 3-9544820 3-9549043
3-9550197 3-9550383 3-9550920 3-9559551 3-9562678 3-9564265 3-9567387 3-9567513 3-9571449
3-9573832 3-9576347 3-9577805 3-9578248 3-9581180 3-9581355 3-9581819 3-9585399 3-9586845

3-9588803 3-9591076 3-9591875 3-9592988 3-9594360 3-9603871 3-9606557 3-9610316 3-9617267 3-9617763 3-9619676 3-9630533 3-9633138 3-9641387 3-9648368 3-9648857 3-9651673 3-9651857 3-9655375 3-9664165 3-9664278 3-9664574 3-9665828 3-9668646 3-9671332 3-9674996 3-9677788 3-9686556 3-9696968 3-9697093 3-9699030 3-9699128 3-9702846 3-9706261 3-9706472 3-9706476 3-9707754 3-9721875 3-9726756 3-9729958 3-9729999 3-9730805 3-9735517 3-9739722 3-9747510 3-9748987 3-9763317 3-9773472 3-9781047 3-9781155 3-9781594 3-9784061 3-9787243 3-9791489 3-9791720 3-9792786 3-9793526 3-9793996 3-9794718 3-9808224 3-9809925 3-9825037 3-9829395 3-9831136 3-9831659 3-9834415 3-9837263 3-9837470 3-9837915 3-9838337 3-9845941 3-9847376 3-9847837 3-9848830 3-9848859 3-9849920 3-9851152 3-9855890 3-9856240 3-9858054 3-9860067 3-9863618 3-9863914 3-9874904 3-9879396 3-9879616 3-9882895 3-9885054 3-9891044 3-9891399 3-9891412 3-9892453 3-9896997 3-9899367 3-9901128 3-9901367 3-9908060 3-9914817 3-9914910 3-9915196 3-9915923 3-9924343 3-9927307 3-9928910 3-9929068 3-9929146 3-9929847 3-9935629 3-9940767 3-9949408 3-9953464 3-9956092 3-9973331 3-9977602 3-9979896 3-9980896 3-9981855 3-9983712 3-9987284 3-9998670

Dataset: Yle 2

20-100929 20-101113 20-101500 20-101577 20-102407 20-103154 20-103574 20-105040 20-106831 20-107055 20-109756 20-113164 20-113555 20-114718 20-115975 20-115991 20-116012 20-116446 20-116853 20-117484 20-118091 20-118151 20-118597 20-119340 20-120124 20-120388 20-120412 20-162485 20-162517 20-216881 20-218690 20-220702 20-227746 20-232381 20-235215 20-236094 20-236660 20-236876 20-243309 20-246013 20-246935 20-248203 20-248697 20-249853 20-249991 20-250599 20-250960 20-30754 20-30762 20-34888 20-34930 20-41431 20-42035 20-55654 20-55792 20-55914 20-56161 20-56346 20-56476 20-56477 20-56510 20-56710 20-56850 20-56983 20-57001 20-57957 20-58713 20-58739 20-60211 20-60379 20-61923 20-63173 20-63419 20-63601 20-63662 20-65019 20-67968 20-69137 20-69734 20-73687 20-74155 20-74442 20-79319 20-79634 20-81405 20-82172 20-82207 20-82384 20-82479 20-82553 20-82571 20-82637 20-82681 20-82686 20-82724 20-84165 20-84191 20-84714 20-85750 20-86569 20-86868 20-88233 20-88236 20-89285 20-89606 20-89872 20-90495 20-93078 20-93874 20-94125 20-95003 20-95869 20-96011 20-97734 20-98935 20-99350 20-99353 20-99366 20-99570 3-5052495 3-5052568 3-5052596 3-5052666 3-5052800 3-5052912 3-5052924 3-5052928 3-5052943 3-5052949 3-5052950 3-5052952 3-5052955 3-5053168 3-5053208 3-5053255 3-5053309 3-5053351 3-5054063 3-5054502 3-5054503 3-5054507

3-5054529 3-5054531 3-5054572 3-5054582 3-5054592 3-5054766 3-5054768 3-5054769 3-5054771
3-5054818 3-5055160 3-5055765 3-5056331 3-5057873 3-5059577 3-5063263 3-5064029 3-5064971
3-5067507 3-5071771 3-5073016 3-5077880 3-5078076 3-5078108 3-5079129 3-5079367 3-5079516
3-5081323 3-5083616 3-5092561 3-5092707 3-5093000 3-5093675 3-5093755 3-5094133 3-5094465
3-5095817 3-5097547 3-5098267 3-5236136 3-5236142 3-5236147 3-5236151 3-5236160 3-5236604
3-5236674 3-5239295 3-5239364 3-5241305 3-5241311 3-5241315 3-5241350 3-5241440 3-5248408
3-5252212 3-5260670 3-5284527 3-5285124 3-5285145 3-5285170 3-5291195 3-5291405 3-5291531
3-5291834 3-5292130 3-5292260 3-5292481 3-5295473 3-5299185 3-5299427 3-5299775 3-5299798
3-5301741 3-5307417 3-5307504 3-5307727 3-5311863 3-5311929 3-5317502 3-5317511 3-5317515
3-5317516 3-5341971 3-5342817 3-5368443 3-5380928 3-5384555 3-5405704 3-5405848 3-5407569
3-5421675 3-5430860 3-5432856 3-5437588 3-5439551 3-5471836 3-5488589 3-5493231 3-5493235
3-5493246 3-5503022 3-5504026 3-5508295 3-5508300 3-5530778 3-5534645 3-5539609 3-5545809
3-5546042 3-5555945 3-5580199 3-5584185 3-5609068 3-5614011 3-5621360 3-5625243 3-5626329
3-5627099 3-5630176 3-5631880 3-5633637 3-5652312 3-5652361 3-5654469 3-5666622 3-5685194
3-5686891 3-5690254 3-5690985 3-5691277 3-5691661 3-5711151 3-5711710 3-5712867 3-5713991
3-5714012 3-5714311 3-5715608 3-5716937 3-5730083 3-5738791 3-5741676 3-5742233 3-5742747
3-5742856 3-5742913 3-5743085 3-5743835 3-5743884 3-5743892 3-5743906 3-5765569 3-5765570
3-5766270 3-5766275 3-5766864 3-5767088 3-5767527 3-5767886 3-5768454 3-5768818 3-5769147
3-5770557 3-5773254 3-5773262 3-5773336 3-5773348 3-5773933 3-5773953 3-5774501 3-5774559
3-5775075 3-5775328 3-5775597 3-5775713 3-5776013 3-5853601 3-5853621 3-5853705 3-5853774
3-5853787 3-5853810 3-5853881 3-5859463 3-5864773 3-5872225 3-5875896 3-5875907 3-5876119
3-5879918 3-5890635 3-5899499 3-5900870 3-5916280 3-5921007 3-5990676 3-5997759 3-6001120
3-6006186 3-6006511 3-6007035 3-6007275 3-6007390 3-6007490 3-6007569 3-6007855 3-6007971
3-6008034 3-6008180 3-6008251 3-6008490 3-6008709 3-6009060 3-6009213 3-6009483 3-6009792
3-6009800 3-6015845 3-6020081 3-6021731 3-6072153 3-6072343 3-6072371 3-6072458 3-6072526
3-6072829 3-6072980 3-6073030 3-6073079 3-6073466 3-6073909 3-6073910 3-6073940 3-6074131
3-6074154 3-6074365 3-6074590 3-6074699 3-6074807 3-6074848 3-6074943 3-6075065 3-6075510
3-6075622 3-6076060 3-6076649 3-6077325 3-6077463 3-6078280 3-6078435 3-6078518 3-6078705
3-6078776 3-6078824 3-6079009 3-6079695 3-6080021 3-6080056 3-6080274 3-6080403 3-6080405
3-6080449 3-6080501 3-6080902 3-6081762 3-6081860 3-6082283 3-6082935 3-6082958 3-6083683
3-6083774 3-6084747 3-6084811 3-6085730 3-6085751 3-6085808 3-6086104 3-6086496 3-6086871
3-6087068 3-6087111 3-6088014 3-6088018 3-6088226 3-6088255 3-6088274 3-6088350 3-6088574

3-6089703 3-6089706 3-6089809 3-6089826 3-6089844 3-6089847 3-6090225 3-6090722 3-6090817
3-6091166 3-6091391 3-6091409 3-6091499 3-6091888 3-6092012 3-6092349 3-6094301 3-6094515
3-6094645 3-6094996 3-6095268 3-6095449 3-6095463 3-6095482 3-6095586 3-6095809 3-6095829
3-6096022 3-6096056 3-6096147 3-6096178 3-6096254 3-6096707 3-6096746 3-6097309 3-6097477
3-6098208 3-6098953 3-6099184 3-6099774 3-6100190 3-6100199 3-6101749 3-6101777 3-6101779
3-6101863 3-6102957 3-6103703 3-6103822 3-6104321 3-6104748 3-6104781 3-6104959 3-6104986
3-6105502 3-6105643 3-6105695 3-6106469 3-6106944 3-6107162 3-6107270 3-6107342 3-6108028
3-6108135 3-6108621 3-6108742 3-6109097 3-6109138 3-6109153 3-6109506 3-6111550 3-6131415
3-6134765 3-6134940 3-6135219 3-6135272 3-6135497 3-6135679 3-6135921 3-6135986 3-6136234
3-6136397 3-6137207 3-6137502 3-6138553 3-6138590 3-6139319 3-6139491 3-6139783 3-6140358
3-6140649 3-6141007 3-6143443 3-6143615 3-6143629 3-6144334 3-6144746 3-6145376 3-6145449
3-6145631 3-6145666 3-6145670 3-6145786 3-6145928 3-6146093 3-6146166 3-6146607 3-6147479
3-6147698 3-6147704 3-6147706 3-6147913 3-6148199 3-6148254 3-6148287 3-6148795 3-6148988
3-6149282 3-6152849 3-6154976 3-6160055 3-6160594 3-6160618 3-6160897 3-6161445 3-6161490
3-6161830 3-6163618 3-6165303 3-6165367 3-6166128 3-6166697 3-6167786 3-6168153 3-6168390
3-6169704 3-6170412 3-6170807 3-6171131 3-6171193 3-6171260 3-6171547 3-6172252 3-6172284
3-6174345 3-6174480 3-6174911 3-6174959 3-6174972 3-6175011 3-6175590 3-6176005 3-6176288
3-6176345 3-6176348 3-6176432 3-6177198 3-6177365 3-6177782 3-6178075 3-6178100 3-6179191
3-6179245 3-6179310 3-6179499 3-6179626 3-6179956 3-6180182 3-6180335 3-6180591 3-6180720
3-6180955 3-6181053 3-6181072 3-6181130 3-6181152 3-6181294 3-6181353 3-6181434 3-6181834
3-6182114 3-6182176 3-6182300 3-6182332 3-6182447 3-6182730 3-6183003 3-6183503 3-6183509
3-6183601 3-6183622 3-6184182 3-6184462 3-6184729 3-6184990 3-6185549 3-6185722 3-6185863
3-6186050 3-6186887 3-6186902 3-6187239 3-6187446 3-6188804 3-6188940 3-6189174 3-6189345
3-6189577 3-6189783 3-6190042 3-6190216 3-6191131 3-6191279 3-6191299 3-6191519 3-6191612
3-6191659 3-6191788 3-6192230 3-6192581 3-6193575 3-6194747 3-6195003 3-6195355 3-6195793
3-6195819 3-6196606 3-6197106 3-6197487 3-6197537 3-6198128 3-6198168 3-6198439 3-6198543
3-6198691 3-6200397 3-6200677 3-6200831 3-6201338 3-6201576 3-6203285 3-6203609 3-6203732
3-6203790 3-6203797 3-6204155 3-6204453 3-6204474 3-6204996 3-6205156 3-6205193 3-6205718
3-6206303 3-6206435 3-6206951 3-6207057 3-6207290 3-6207344 3-6207556 3-6207918 3-6207995
3-6209029 3-6209671 3-6211444 3-6211999 3-6212019 3-6212029 3-6212123 3-6212239 3-6212666
3-6212838 3-6213640 3-6214694 3-6215166 3-6215318 3-6216000 3-6216163 3-6216202 3-6216553
3-6216576 3-6217242 3-6219725 3-6219852 3-6220280 3-6220322 3-6220404 3-6220510 3-6221157

3-6221594 3-6221953 3-6222202 3-6222847 3-6222984 3-6223163 3-6223889 3-6224012 3-6224533
3-6224541 3-6224691 3-6225016 3-6225046 3-6226083 3-6226319 3-6226637 3-6226711 3-6226938
3-6227334 3-6227691 3-6227923 3-6228076 3-6228436 3-6229687 3-6229820 3-6230086 3-6230488
3-6230538 3-6230634 3-6230666 3-6230684 3-6230699 3-6230874 3-6230954 3-6231011 3-6231027
3-6231464 3-6231927 3-6232175 3-6232329 3-6232416 3-6232500 3-6233073 3-6233761 3-6234343
3-6234348 3-6235352 3-6235711 3-6235807 3-6235840 3-6236200 3-6236317 3-6236520 3-6236605
3-6237067 3-6238094 3-6238227 3-6238419 3-6239297 3-6239383 3-6239504 3-6239611 3-6239912
3-6240496 3-6240752 3-6240943 3-6240954 3-6241147 3-6241393 3-6241894 3-6243542 3-6243902
3-6243948 3-6244009 3-6244664 3-6244669 3-6245406 3-6245426 3-6245445 3-6245574 3-6246385
3-6247232 3-6247725 3-6248278 3-6248919 3-6249162 3-6249195 3-6249734 3-6250257 3-6250721
3-6250914 3-6251171 3-6251438 3-6251631 3-6254001 3-6255158 3-6255262 3-6255578 3-6255932
3-6256126 3-6256567 3-6257395 3-6257459 3-6258304 3-6258539 3-6258790 3-6258808 3-6258887
3-6258954 3-6259099 3-6259348 3-6260797 3-6261265 3-6262737 3-6262767 3-6262798 3-6263697
3-6265134 3-6265221 3-6265725 3-6265731 3-6265732 3-6266465 3-6266576 3-6266668 3-6266788
3-6267166 3-6268554 3-6268676 3-6269260 3-6270303 3-6271223 3-6271510 3-6273327 3-6273700
3-6273940 3-6273981 3-6274257 3-6275250 3-6275328 3-6275509 3-6276996 3-6277268 3-6277564
3-6278403 3-6279740 3-6280029 3-6280809 3-6281043 3-6281309 3-6281614 3-6281712 3-6281808
3-6282300 3-6282338 3-6282407 3-6283111 3-6283217 3-6283244 3-6286102 3-6286187 3-6286216
3-6287002 3-6288336 3-6288744 3-6289707 3-6290498 3-6291055 3-6291649 3-6292261 3-6292482
3-6292810 3-6293302 3-6294776 3-6294828 3-6294930 3-6295213 3-6297137 3-6297154 3-6297284
3-6297446 3-6298010 3-6298390 3-6298626 3-6299178 3-6299381 3-6301157 3-6301431 3-6301875
3-6302051 3-6302600 3-6302923 3-6303733 3-6303786 3-6304043 3-6304049 3-6304465 3-6304710
3-6306723 3-6306812 3-6308267 3-6308642 3-6308709 3-6309309 3-6309442 3-6309479 3-6309712
3-6310530 3-6311662 3-6311711 3-6311922 3-6312262 3-6312287 3-6312296 3-6312786 3-6312797
3-6313287 3-6313921 3-6316790 3-6318621 3-6318974 3-6319247 3-6319544 3-6319859 3-6320007
3-6320924 3-6321844 3-6322085 3-6322713 3-6322786 3-6322788 3-6322826 3-6322979 3-6323300
3-6325508 3-6325555 3-6325786 3-6326379 3-6327241 3-6327639 3-6327721 3-6328494 3-6329903
3-6330574 3-6331166 3-6331276 3-6331638 3-6332610 3-6333674 3-6335660 3-6335744 3-6335797
3-6336042 3-6336526 3-6336531 3-6336561 3-6336771 3-6336971 3-6337250 3-6337426 3-6338358
3-6339265 3-6339312 3-6339971 3-6342280 3-6342442 3-6342791 3-6344556 3-6345142 3-6345364
3-6345758 3-6345883 3-6346859 3-6346975 3-6347147 3-6347515 3-6348032 3-6348356 3-6348424
3-6348873 3-6349636 3-6349694 3-6349856 3-6350115 3-6350139 3-6350317 3-6354901 3-6355494

3-6355757 3-6356500 3-6359307 3-6359505 3-6360116 3-6360321 3-6362072 3-6363772 3-6364240
3-6364778 3-6364939 3-6365052 3-6365256 3-6365278 3-6367361 3-6367935 3-6368030 3-6368171
3-6368502 3-6368831 3-6369141 3-6369379 3-6370563 3-6370588 3-6372354 3-6372507 3-6372708
3-6373359 3-6374296 3-6374497 3-6374890 3-6375607 3-6375617 3-6375704 3-6376412 3-6377308
3-6377455 3-6377564 3-6377773 3-6378264 3-6379250 3-6379803 3-6380563 3-6382656 3-6382971
3-6382993 3-6383001 3-6383546 3-6384200 3-6384433 3-6384968 3-6385032 3-6385154 3-6386701
3-6386751 3-6387094 3-6387335 3-6387857 3-6390060 3-6390895 3-6391092 3-6391155 3-6392137
3-6392449 3-6392920 3-6393198 3-6394091 3-6394138 3-6394192 3-6394319 3-6395516 3-6395748
3-6395944 3-6395981 3-6396008 3-6396408 3-6396603 3-6397164 3-6397536 3-6399029 3-6400196
3-6400378 3-6400532 3-6400962 3-6401758 3-6401869 3-6401950 3-6402960 3-6403235 3-6403723
3-6405183 3-6405600 3-6406618 3-6407074 3-6407886 3-6408042 3-6409376 3-6409427 3-6409549
3-6410661 3-6410915 3-6411634 3-6413020 3-6413366 3-6413621 3-6413949 3-6414040 3-6414475
3-6414566 3-6414631 3-6415624 3-6416383 3-6416573 3-6417513 3-6418227 3-6418373 3-6418809
3-6419461 3-6420029 3-6420687 3-6420782 3-6421354 3-6421704 3-6422412 3-6422649 3-6423190
3-6423309 3-6423889 3-6424213 3-6424758 3-6425990 3-6426546 3-6426917 3-6427014 3-6428923
3-6429468 3-6429484 3-6430147 3-6430487 3-6430906 3-6432319 3-6432694 3-6434400 3-6434429
3-6434626 3-6434970 3-6435123 3-6435363 3-6435578 3-6435915 3-6435946 3-6436109 3-6436465
3-6437420 3-6437886 3-6438313 3-6439807 3-6440205 3-6440292 3-6440384 3-6440394 3-6440486
3-6440715 3-6441272 3-6441864 3-6441930 3-6442260 3-6442599 3-6442651 3-6443360 3-6443633
3-6444108 3-6444317 3-6444785 3-6445684 3-6446850 3-6447388 3-6448289 3-6449446 3-6449485
3-6449718 3-6449849 3-6450362 3-6450399 3-6450865 3-6450885 3-6451075 3-6451618 3-6453318
3-6453388 3-6453841 3-6454076 3-6454220 3-6454830 3-6455056 3-6455059 3-6455098 3-6455414
3-6455672 3-6456878 3-6456906 3-6457052 3-6457110 3-6457370 3-6458118 3-6458300 3-6458887
3-6459797 3-6460032 3-6460112 3-6461190 3-6461301 3-6461759 3-6463081 3-6463196 3-6463198
3-6465386 3-6465539 3-6466550 3-6466633 3-6466679 3-6466793 3-6467193 3-6467497 3-6468935
3-6469292 3-6470334 3-6470569 3-6470936 3-6470992 3-6471030 3-6471048 3-6471943 3-6472550
3-6473748 3-6473793 3-6474274 3-6475095 3-6476734 3-6478332 3-6479634 3-6480043 3-6480690
3-6481096 3-6481550 3-6481973 3-6482018 3-6483436 3-6484088 3-6484841 3-6486784 3-6489377
3-6490089 3-6490997 3-6491882 3-6491942 3-6492307 3-6492943 3-6493220 3-6494050 3-6496301
3-6496613 3-6497402 3-6498003 3-6498610 3-6499332 3-6499730 3-6500263 3-6500284 3-6500376
3-6500605 3-6501088 3-6501548 3-6501751 3-6501960 3-6502899 3-6503143 3-6503911 3-6505961
3-6507039 3-6508858 3-6510210 3-6513230 3-6514893 3-6515331 3-6516585 3-6516731 3-6517224

3-6517395 3-6517512 3-6518977 3-6519688 3-6519720 3-6520836 3-6521025 3-6522000 3-6522009
3-6524123 3-6525238 3-6525538 3-6525946 3-6527191 3-6527385 3-6527397 3-6528683 3-6528705
3-6531748 3-6532515 3-6532725 3-6533474 3-6534242 3-6534520 3-6535145 3-6535464 3-6535489
3-6535880 3-6537361 3-6537630 3-6538119 3-6539488 3-6539672 3-6541536 3-6542785 3-6543086
3-6544168 3-6545226 3-6545507 3-6545885 3-6546104 3-6548050 3-6548324 3-6549486 3-6549827
3-6550000 3-6550087 3-6551181 3-6552833 3-6553140 3-6554317 3-6556350 3-6556626 3-6557178
3-6557240 3-6560808 3-6560845 3-6561057 3-6561701 3-6562273 3-6562600 3-6563904 3-6565035
3-6566150 3-6566385 3-6566947 3-6567746 3-6569100 3-6569695 3-6569741 3-6569742 3-6569875
3-6570751 3-6572034 3-6573354 3-6573427 3-6573583 3-6573797 3-6575538 3-6576356 3-6576406
3-6578266 3-6578915 3-6580090 3-6580145 3-6580843 3-6581681 3-6583923 3-6584827 3-6586011
3-6587432 3-6590387 3-6590520 3-6591387 3-6591575 3-6591713 3-6591806 3-6592616 3-6606840
3-6606947 3-6611493 3-6615336 3-6617194 3-6618276 3-6624047 3-6628020 3-6634552 3-6634933
3-6636017 3-6636485 3-6638455 3-6638674 3-6638950 3-6640436 3-6640549 3-6641137 3-6641393
3-6642504 3-6644603 3-6647796 3-6648462 3-6648912 3-6650063 3-6650368 3-6650532 3-6651179
3-6651518 3-6651564 3-6652671 3-6652896 3-6653020 3-6653428 3-6654092 3-6655261 3-6655537
3-6657022 3-6658354 3-6658368 3-6658668 3-6660831 3-6662313 3-6662347 3-6662532 3-6663447
3-6664444 3-6666049 3-6666888 3-6666889 3-6666929 3-6667038 3-6667140 3-6667209 3-6668115
3-6670886 3-6672074 3-6672304 3-6672950 3-6673416 3-6674204 3-6675938 3-6675998 3-6676410
3-6678922 3-6682759 3-6683527 3-6683687 3-6685961 3-6689455 3-6691170 3-6691858 3-6691894
3-6692045 3-6692441 3-6693553 3-6693663 3-6694273 3-6696990 3-6698532 3-6699964 3-6699966
3-6700825 3-6704126 3-6705958 3-6706345 3-6706380 3-6706513 3-6706555 3-6706562 3-6706720
3-6706907 3-6707295 3-6707430 3-6707937 3-6709560 3-6713417 3-6714425 3-6714522 3-6714557
3-6716628 3-6720175 3-6720378 3-6721765 3-6723836 3-6724698 3-6725076 3-6726075 3-6727278
3-6727572 3-6730256 3-6731998 3-6732081 3-6733791 3-6734778 3-6735938 3-6736679 3-6736713
3-6742634 3-6744915 3-6746262 3-6747938 3-6749114 3-6749702 3-6749757 3-6750101 3-6751363
3-6752413 3-6755882 3-6758112 3-6761711 3-6767197 3-6767902 3-6768131 3-6768316 3-6768714
3-6769262 3-6769457 3-6769480 3-6770361 3-6770843 3-6772967 3-6773991 3-6774381 3-6774492
3-6774711 3-6775816 3-6777069 3-6777240 3-6777562 3-6778386 3-6778559 3-6778706 3-6778747
3-6778937 3-6783459 3-6785069 3-6785298 3-6785319 3-6785383 3-6788381 3-6789256 3-6790781
3-6794418 3-6795305 3-6797162 3-6797285 3-6809768 3-6810289 3-6813600 3-6816960 3-6817920
3-6819621 3-6820276 3-6820907 3-6821598 3-6821726 3-6822906 3-6824359 3-6825787 3-6828373
3-6829242 3-6829617 3-6830644 3-6832613 3-6836737 3-6836920 3-6837136 3-6839933 3-6841044

3-6844733 3-6845137 3-6846253 3-6848179 3-6848898 3-6849699 3-6850436 3-6850514 3-6850907
3-6852047 3-6853715 3-6855499 3-6857112 3-6859190 3-6862125 3-6865598 3-6869219 3-6873605
3-6873610 3-6875567 3-6876673 3-6877007 3-6880387 3-6880564 3-6882484 3-6882486 3-6884905
3-6888669 3-6889273 3-6889277 3-6889286 3-6889588 3-6892740 3-6894717 3-6895508 3-6898022
3-6899527 3-6900586 3-6903133 3-6903815 3-6904387 3-6906251 3-6907994 3-6908878 3-6910702
3-6912388 3-6912460 3-6912714 3-6913238 3-6913331 3-6915288 3-6918000 3-6918470 3-6918501
3-6918733 3-6918770 3-6919427 3-6920027 3-6920146 3-6920331 3-6921074 3-6921096 3-6921100
3-6921654 3-6921823 3-6927342 3-6927414 3-6928169 3-6929354 3-6929543 3-6929904 3-6930100
3-6931612 3-6931795 3-6931873 3-6932155 3-6934822 3-6934889 3-6935477 3-6936203 3-6936229
3-6936407 3-6936447 3-6938844 3-6941521 3-6942630 3-6942851 3-6943686 3-6943863 3-6944067
3-6944074 3-6945777 3-6945858 3-6947063 3-6948177 3-6949441 3-6952746 3-6953158 3-6954114
3-6955714 3-6955980 3-6957007 3-6958013 3-6959269 3-6959474 3-6960158 3-6960627 3-6960665
3-6960853 3-6961690 3-6963552 3-6963875 3-6963958 3-6964351 3-6965123 3-6966034 3-6966709
3-6968100 3-6970026 3-6971611 3-6971849 3-6972108 3-6972230 3-6974266 3-6977770 3-6979298
3-6980601 3-6981056 3-6981279 3-6981440 3-6981785 3-6982353 3-6984423 3-6987409 3-6987647
3-6989265 3-6992055 3-6992509 3-6992794 3-6992941 3-6992993 3-6994440 3-6995206 3-6998776
3-7001867 3-7004723 3-7007119 3-7008196 3-7008876 3-7009527 3-7010846 3-7013113 3-7021132
3-7021144 3-7022065 3-7023612 3-7024349 3-7025758 3-7025993 3-7027307 3-7027748 3-7031390
3-7031392 3-7032382 3-7033942 3-7034869 3-7037648 3-7037780 3-7038261 3-7039374 3-7039822
3-7041338 3-7042145 3-7044039 3-7044109 3-7051166 3-7054096 3-7054476 3-7056681 3-7060528
3-7060850 3-7061728 3-7066219 3-7066246 3-7066996 3-7068132 3-7068329 3-7068562 3-7069973
3-7071076 3-7071423 3-7071500 3-7073542 3-7074514 3-7074782 3-7074859 3-7078973 3-7080098
3-7080561 3-7080706 3-7081090 3-7081549 3-7082894 3-7083405 3-7088985 3-7092361 3-7093786
3-7094027 3-7094702 3-7096455 3-7102130 3-7103992 3-7104558 3-7104820 3-7105024 3-7106227
3-7108733 3-7111770 3-7114790 3-7114952 3-7116624 3-7117808 3-7118335 3-7123258 3-7123315
3-7123537 3-7127238 3-7127502 3-7127588 3-7128700 3-7131402 3-7131734 3-7132724 3-7132898
3-7133154 3-7133181 3-7134916 3-7135341 3-7135716 3-7137030 3-7137051 3-7137293 3-7141202
3-7142123 3-7142260 3-7142608 3-7143256 3-7143533 3-7144342 3-7144520 3-7145723 3-7146240
3-7149470 3-7152893 3-7156445 3-7156469 3-7157085 3-7157239 3-7157492 3-7157562 3-7159718
3-7160310 3-7161642 3-7161845 3-7163565 3-7168557 3-7170308 3-7171187 3-7171309 3-7171593
3-7173478 3-7183559 3-7185694 3-7190793 3-7192862 3-7194889 3-7199365 3-7200460 3-7208798
3-7209635 3-7215013 3-7216865 3-7223738 3-7225138 3-7227640 3-7230581 3-7236953 3-7242080

3-7242857 3-7248191 3-7249989 3-7257714 3-7258713 3-7259650 3-7261050 3-7268504 3-7272373 3-7278926 3-7299324 3-7299452 3-7300019 3-7303121 3-7303622 3-7305335 3-7317317 3-7323944 3-7327284 3-7327896 3-7338282 3-7338295 3-7341144 3-7348497 3-7350564 3-7352824 3-7359320 3-7366227 3-7368688 3-7375702 3-7378934 3-7383913 3-7385250 3-7385273 3-7390214 3-7392995 3-7393168 3-7396771 3-7399068 3-7405549 3-7405628 3-7409595 3-7416128 3-7422500 3-7434727 3-7440054 3-7443983 3-7457779 3-7458779 3-7465348 3-7467400 3-7470375 3-7470921 3-7476802 3-7481449 3-7484129 3-7484296 3-7485215 3-7486817 3-7487020 3-7495421 3-7499449 3-7502829 3-7502917 3-7515413 3-7531420 3-7531936 3-7536252 3-7539414 3-7539529 3-7550404 3-7555194 3-7574114 3-7595422 3-7605988 3-7609756 3-7613027 3-7640047 3-7640165 3-7640643 3-7641050 3-7641629 3-7646029 3-7664961 3-7665257 3-7667657 3-7674219 3-7678882 3-7687363 3-7706324 3-7713690

6.1 Dataset: Finlex

http://data.finlex.fi/eli/sd/1734/4/luku/17/pykala/22/ajantasa/20160101 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1734/4/luku/2/pykala/8/ajantasa/20190101 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1734/4/luku/8/pykala/4/ajantasa/20090101 perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/1889/39/luku/12/pykala/11/ajantasa/19950901 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1889/39/luku/45/pykala/3/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1889/39/luku/8/pykala/2/ajantasa/20060101 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1929/234/osa/1/luku/4/pykala/16/ajantasa/19880101 perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/1946/436/pykala/5/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/1965/40/luku/20/pykala/4/ajantasa/20080101 perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/1966/91/pykala/3/ajantasa/20101201 omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/1968/360/osa/3/luku/1/pykala/27b/ajantasa/19880701 verotus

http://data.finlex.fi/eli/sd/1972/666/pykala/11/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1981/418/luku/4/pykala/20/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/1985/314/luku/4/pykala/29/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/1987/355/luku/8/pykala/43/alkup omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/1987/395/luku/5/pykala/35a/ajantasa/20090101 omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/1989/354/pykala/2/alkup julkishallinto valtionhallinto

http://data.finlex.fi/eli/sd/1992/1257/luku/2/pykala/5/ajantasa/19921204 liikenne kuljetus

http://data.finlex.fi/eli/sd/1993/1054/luku/6/pykala/41/ajantasa/19931126 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/1993/47/luku/3/pykala/10/ajantasa/20070601 rahoitus

http://data.finlex.fi/eli/sd/1993/770/pykala/2/ajantasa/20180101 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/1994/1072/luku/6/pykala/44/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/1994/674/luku/16/pykala/12/ajantasa/19940715 rahoitus

http://data.finlex.fi/eli/sd/1994/719/luku/3/pykala/11d/ajantasa/20190101 liikenne kuljetus

http://data.finlex.fi/eli/sd/1994/843/luku/6/pykala/24/ajantasa/20190101 omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/1995/1710/luku/1/pykala/1/alkup liikenne kuljetus

http://data.finlex.fi/eli/sd/1995/540/luku/16/pykala/4a/ajantasa/19950412 asuminen kiinteistö

http://data.finlex.fi/eli/sd/1995/554/luku/3/pykala/14/ajantasa/20140101 asuminen kiinteistö

http://data.finlex.fi/eli/sd/1998/986/luku/7/pykala/22a/ajantasa/20040101 julkishallinto valtionhallinto

http://data.finlex.fi/eli/sd/1999/1084/pykala/14/ajantasa/20111231 rahoitus

http://data.finlex.fi/eli/sd/1999/731/luku/2/pykala/23/ajantasa/20120301 ihmisoikeudet perusoikeudet

http://data.finlex.fi/eli/sd/2002/1290/luku/5/pykala/12/ajantasa/20110801 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2002/194/luku/5/pykala/26/ajantasa/20190101 liikenne kuljetus

http://data.finlex.fi/eli/sd/2002/241/luku/8/pykala/32/alkup omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/2002/257/luku/2/pykala/10/ajantasa/20091201 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/2003/1197/luku/10/pykala/46/alkup julkishallinto valtionhallinto

http://data.finlex.fi/eli/sd/2003/1197/luku/6/pykala/27/alkup julkishallinto valtionhallinto

http://data.finlex.fi/eli/sd/2003/1286/luku/2/pykala/31/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/2003/304/luku/8/pykala/37b/ajantasa/20170101 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2004/1224/luku/12/pykala/2/ajantasa/20170101 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2004/629/luku/3/pykala/17/ajantasa/20100101 liikenne kuljetus

http://data.finlex.fi/eli/sd/2005/1142/luku/4/pykala/21/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/2005/767/luku/18/pykala/9/ajantasa/20150501 rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/s/2006/1233/pykala/6/ajantasa/20130101 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2006/1280/luku/5/pykala/59e/ajantasa/20170101 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2006/624/luku/3/pykala/1/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2006/624/luku/5/pykala/9/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2007/273/luku/2/pykala/20d/ajantasa/20160101 liikenne kuljetus

http://data.finlex.fi/eli/sd/2007/370/luku/5/pykala/27/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/2007/417/luku/3/pykala/13a/ajantasa/20100301 perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/2007/705/luku/5/pykala/27/alkup omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/2009/228/pykala/4/ajantasa/20140701 liikenne kuljetus

http://data.finlex.fi/eli/sd/2009/299/pykala/10/alkup verotus

http://data.finlex.fi/eli/sd/2010/934/luku/5/pykala/34/alkup rahoitus

http://data.finlex.fi/eli/sd/2011/210/pykala/3/alkup liikenne kuljetus

http://data.finlex.fi/eli/sd/2011/379/luku/2/pykala/6/ajantasa/20110429 ympäristö

http://data.finlex.fi/eli/sd/2011/386/luku/7/pykala/68/ajantasa/20190601 liikenne kuljetus

http://data.finlex.fi/eli/sd/2011/439/luku/1/pykala/4/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/2011/646/luku/12/pykala/114/alkup ympäristö

http://data.finlex.fi/eli/sd/2011/715/pykala/20/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/2011/756/luku/2/pykala/3/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2011/756/luku/8/pykala/11/ajantasa/20170101 yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2012/100/luku/9/pykala/65/ajantasa/20190101 julkishallinto valtionhallinto

http://data.finlex.fi/eli/sd/2012/22/luku/10/pykala/86/alkup perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/2012/22/luku/2/pykala/3/alkup perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/2012/391/luku/1/pykala/3/alkup ympäristö

http://data.finlex.fi/eli/sd/2013/1201/luku/2/pykala/6/alkup omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/2013/781/pykala/2/alkup verotus

http://data.finlex.fi/eli/sd/2014/1174/luku/4/pykala/19/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/2014/416/luku/4/pykala/84/alkup julkishallinto valtionhallinto

http://data.finlex.fi/eli/sd/2014/527/luku/16/pykala/154/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/2014/527/luku/21/pykala/237/alkup ympäristö

http://data.finlex.fi/eli/sd/2015/1083/pykala/2/alkup ympäristö

http://data.finlex.fi/eli/sd/2015/11/luku/3/pykala/18/alkup perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/2015/1426/pykala/1/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/2015/1696/pykala/8/alkup omaisuus kaupankäynti kuluttajansuoja

http://data.finlex.fi/eli/sd/2015/216/pykala/11/alkup asuminen kiinteistö

http://data.finlex.fi/eli/sd/2015/327/luku/4/pykala/17/ajantasa/20170316 ympäristö

http://data.finlex.fi/eli/sd/2015/400/luku/3/pykala/10/alkup rikosasiat oikeudenkäynti

http://data.finlex.fi/eli/sd/2015/685/luku/3/pykala/9/alkup ympäristö

http://data.finlex.fi/eli/sd/2015/80/luku/4/pykala/17/alkup rahoitus

http://data.finlex.fi/eli/sd/2016/304/luku/3/pykala/12/alkup liikenne kuljetus

http://data.finlex.fi/eli/sd/2016/549/luku/5/pykala/40/alkup omaisuus kaupankäynti kulutta-jansuoja

http://data.finlex.fi/eli/sd/2016/734/luku/4/pykala/15/alkup rahoitus

http://data.finlex.fi/eli/sd/2016/768/luku/4/pykala/22/alkup verotus

http://data.finlex.fi/eli/sd/2016/872/pykala/6/alkup koulutus

http://data.finlex.fi/eli/sd/2017/320/osa/5/luku/2/pykala/5/ajantasa/20190101 liikenne kuljetus

http://data.finlex.fi/eli/sd/2017/907/pykala/14/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2017/946/luku/3/pykala/22/alkup perheoikeus perintöoikeus

http://data.finlex.fi/eli/sd/2018/11/luku/4/pykala/29/alkup verotus

http://data.finlex.fi/eli/sd/2018/1302/luku/21/pykala/162/alkup liikenne kuljetus

http://data.finlex.fi/eli/sd/2018/714/luku/10/pykala/63/alkup koulutus

http://data.finlex.fi/eli/sd/2018/729/luku/6/pykala/163/alkup liikenne kuljetus

http://data.finlex.fi/eli/sd/2019/213/luku/20/pykala/8/alkup rahoitus

http://data.finlex.fi/eli/sd/2019/213/luku/6/pykala/1/alkup yritykset yhteisöt työelämä

http://data.finlex.fi/eli/sd/2019/213/luku/9/pykala/10/ajantasa/20190610 rahoitus

http://data.finlex.fi/eli/sd/2019/590/luku/1/pykala/9/alkup ihmisoikeudet perusoikeudet

http://data.finlex.fi/eli/sd/2019/650/luku/2/pykala/8/alkup ihmisoikeudet perusoikeudet

http://data.finlex.fi/eli/sd/2019/782/luku/7/pykala/113/alkup liikenne kuljetus

http://data.finlex.fi/eli/sd/2019/808/luku/2/pykala/9/alkup julkishallinto valtionhallintod