Something something

Clemens Westrup

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 16.1.2015

Thesis supervisor:

Prof. Aristides Gionis

Thesis advisor:

Clemens Westrup

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Clemens Westrup

Title: Something something

Date: 16.1.2015 Language: English Number of pages: 6+42

Department of Information and Computer Science

Professorship: Machine Learning, Data Mining, and Probabilistic Modeling

Supervisor: Prof. Aristides Gionis

Advisor: Clemens Westrup

Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained. Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained.
Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained. Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained.

Keywords: NLP, bla bla, keyword

Preface

I want to thank bla bla bla

New York, 16.1.2015

Clemens Westrup

Contents

Symbols and abbreviations*

Abbreviations*

kNN	k-Nearest Neighbors
SVM	Support Vector Machine
NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient, see Section 2.5.1

Glossary*

one-hot-encoding	TODO
grid search	TODO
Crowdflower	TODO
Mturk	see *Mechanical Turk*
Mechanical Turk	TODO
API	TODO
MongoDB	TODO
Mongoose	TODO
GitHub	TODO

Todo list

1 Introduction

Language is one of the most complex behaviors our species has developed. Humans use it to communicate even the most abstract concepts and it is considered one of the pillars of modern civilization. It takes children years to learn to communicate their thoughts and the subtle nuances of one's language give a glimpse one's cultural environment and upbringing, one's emotional state and one's intellect.

Not surprisingly in the field of Artificial Intelligence (AI) building computer systems with linguistic capabilities and solving language-based problems poses one of the hardest challenges and has motivated decades of research in Computational Linguistics. In fact many of the famous test for universal machine intelligence are based on linguistic capabilities, among them the famous *Turing test* by [Turing, 1950] where the task is for a human judge to determine whether he is having a conversation with a human or a machine in order to determine if the machine can be called intelligent, or the *compression test* proposed by [Mahoney, 1999], where a human's and machine's capability to predict missing words given a context is tested.

This thesis explores the specific task of predicting the semantic structure of job advertisements as a specific example of such a language-based task that turns out to be difficult even for humans to do. The work was done in close collaboration with the Helsinki-based media and learning company *Sanoma*[1] and the research motivation was thus constantly tied back into real world challenges in the scope of Sanoma's business needs.

1.1 Need Statement and Motivation

Today's media and education, the basis of Sanoma's core businesses, are undergoing drastic and fundamental transformations that are currently disrupting whole industries.

Usage of digital media as a source of information has long surpassed print media. Sanoma's most well-known product, Finland's biggest daily newspaper *Helsingin Sanomat*, lost 6% of its circulation only in 2015[2], while the wide-spread use of social media challenges traditional ways we access information. Similarly in the field of education, with the rise of Massive open online course (MOOCs), traditional learning settings are challenged and the need for advanced techniques for data processing and analysis increases, e.g. to personalize and adapt the learning experience to each individual user and at the same time identify trends across large groups of learners to better meet the needs of education.

Sanoma provides a recruitment platform named *Oikotie Työpaikat*. The service is in direct competition several other international players in the recruitment industry.

[1] "Sanoma is a front running consumer media and learning company in Europe. In Finland and the Netherlands we are the market leading media company with a broad presence across multiple platforms. In Belgium we are among the Top 5. Our main markets in learning are Belgium, Finland, the Netherlands, Poland and Sweden. We entertain, inform, educate and inspire millions of people every day. We employ some 7,500 professional employees operating in Europe.", Source: http://www.sanoma.com/en/who-we-are, visited 06.06.2016

[2] Source: http://www.digitalnewsreport.org/survey/2016/finland-2016/, visited 27.07.2016

Through this and other services Sanoma's collects large amounts of user-generated data, offering the potential to be leveraged for machine learning solutions to provide value for their users and innovate and enrich the company's offerings. This was the company's initial motivation for this thesis project — To explore ways to leverage user-generated data to potentially.

From my perspective this offered many interesting research possibilities while at the same time being relevant for a real business. Natural Language Processing and Computer Linguistics had always been of strong interest to me for the complex nature and yet high interpretability of problems and their proximity and relatedness to progress in universal machine intelligence. This presented me with the challenge position to balance pursuing research objectives and yet exploring potential business and user needs, learn on new fronts and deepen my knowledge in others as well as combining my experience in Product Innovation, thus made for a great project to mark the competition of my studies as a Master's student.

1.2 Problem Statement

The problem addressed during with this thesis to better understand the structure of job advertisements. In particular job postings typically consist of several parts with a certain function or theme: Usually company is introduced, the job is described with it's tasks and responsibilities, the requirements for the job are listed, then benefits and offerings by the company are named and the reader is asked to apply in a specified way he or she is interested. Almost all of the text[3] of a job description falls into these categories and the task can thus be posed as predicting a category for each sentence in a job advertisement, that corresponds with this sentence belonging to one of the job ads' parts as described above. This is a challenging problem in itself but can further be used to extract certain functional parts of each job ad, to study a possible correlation between structural patterns and the reach and success of an ad and so forth. The problem therefore can be labelled as *text categorization* or *text classification* as referred to in the scientific literature.

link section

1.3 Research Objectives and Scope

- study vector space models and new approaches - study sequential and multi-task approaches - mu

1.4 Related work *

Algorithmic text categorization into a fixed set of categories has been of a topic of interest for decades.

Boosted by the increasingly vast amounts of data available today. The applications are various, from document filtering, automated metadata generation such as language

[3] Only 4% of the sentences collected for evaluating the final experiments in this thesis were sorted into the category *other* while the rest falls into either of the categories described. This is described in more detail in Section

classification to automatic email labeling, spam identification and sentiment detection, amongst others.

Unsupervised techniques for topic discovery have been investigated widely, such as LSA

Vector Space models are a

- feature learning for text - multitask learning

[Collobert and Weston, 2008] showed how both multitask learning and semi-supervised learning improve the generalization of the shared tasks on text data. They describe "a single convolutional neural network architecture that, given a sentence, outputs [...] part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words and the likelihood that the sentence makes sense (grammatically and semantically) using a language model".

[Lodhi et al., 2002] string kernels

section on transfer learning and feature learning

text classification

Multitask learning

explicit vs implicit feature representation

1.5 Structure of the thesis*

The first section of this thesis gave a brief introduction into the topic of this work, outlined the motivation and the research problem approached and showed the research objectives, the scope of the thesis as well as related work.

Section 2: Background introduces the reader to concepts and ideas of Text Classification in order to provide him or her with the necessary knowledge to understand the work described in this thesis. First the problem of Text Classification is formally defined and the most common approaches to this problem are described on a high level. Afterwards Vector Space Models are introduced in detail, which represent a popular way of tackling this task by transforming text into fixed-size vectors. The following subsection then briefly presents several of the most known classification algorithms that can operate on the vectors produced by such Vector Space Models. Next the approach of Sequential Classification is described in short where text is treaded as a one-dimensional signal in time. The following subsection then shows common ways to evaluate how well the task of text classification is solved and the last subsection gives a brief introduction to two methods that are useful for exploring different models and techniques through visualization.

Section 3: Exploration gives an overview of the exploration of the wider topic space at the start of the thesis project as well as the experimentation with different techniques and the data given for the project. It aims to lay out to the reader the process, insights and learnings leading towards the final problem formulation and evaluation of methods to solve this problem.

The following Section **??**: **??** then presents the main results of the thesis. The exact problem definition is given that is used as a basis to evaluate the experiments and the dataset used for these experiments is described. Subsequently the evaluation of the different approaches to Vector Space Models, the classification algorithms using the most successful of these models and the sequential modeling approach are documented. Each of these subsections first describes the experimental setup, followed by results of the different approaches and closes with a brief discussion on

the results presented. Then

2 Background

This chapter will provide the necessary background on text classification, assuming the reader is familiar with the basic concepts of Machine Learning and related fields. First the research problem will be defined. Then

2.1 Text classification

2.1.1 Problem Formalism

Text classification, also known as text categorization, is the task of predicting a *mapping* $\tilde{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{True, False\}$ between a set of documents $\mathcal{D}$ and a set of classes or categories $\mathcal{C}$ using a model function $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{True, False\}$. This means that were are trying to predict as good as possible the categories that each document belongs into, or vice versa the documents associated with each category. This mapping can thus be represented as a bipartite graph between the sets of documents $\mathcal{D}$ and categories $\mathcal{C}$ as shown in Figure 1. In this representation vertices in the graph indicate a $True$ value in the mapping, indicating that the document and category are associated with each other, while missing vertices indicate that they are not ($False$).

Categories $\mathcal{C}$ are given as symbolic labels and documents $\mathcal{D}$ as chunks of text with variable length. We usually assume that no additional information such as metadata or other *exogenous knowledge* is available on neither labels nor documents. As [Sebastiani, 2002] points out a consequence of relying solely on *endogenous knowledge*, especially the semantics of a text, is that there is no objective ground truth to this task in most settings since semantics are a *subjective* notion: "This is exemplified by the phenomenon of inter-indexer inconsistency [Cleverdon 1984]: when two human experts decide whether to classify document d_j under category c_i, they may disagree, and this in fact happens with relatively high frequency. A news article on Clinton attending Dizzy Gillespie's funeral could be filed under Politics, or under Jazz, or under both, or even under neither, depending on the subjective judgment of the expert." [Sebastiani, 2002]

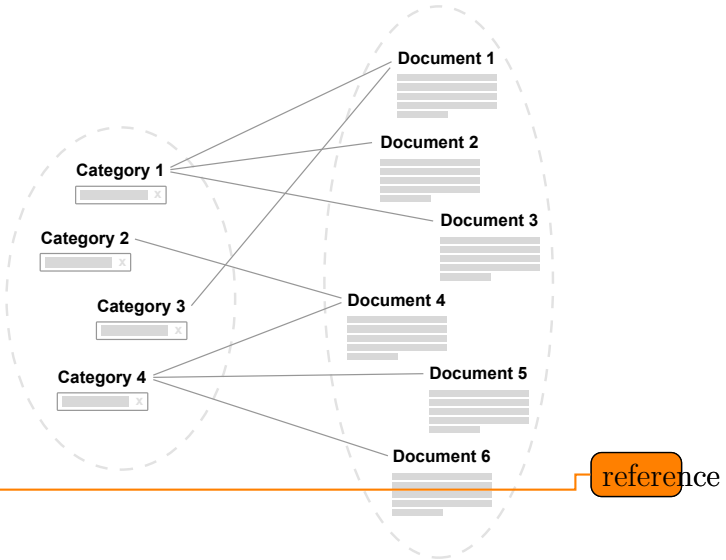


Figure 1: Text classification visualized as a bipartite graph. Here the multi-label setting is shown where no additional constraints are enforced on the problem and hence each document can be assigned to multiple categories

Additional constraints can be imposed on the problem to adapt it for

different application scenarios. Firstly text classification can be either framed as *single-label* classification where each document is assigned to only one single category or *multi-label* classification where an assignment to several categories or also no category is possible. The multi-label case can also be formulated as $|\mathcal{C}|$ individual binary classification problems which of course assumes statistical independence between these prediction tasks.

In order to measure how successfully we are tackling the problem of text classification we need metrics that measure the effectiveness of our algorithm given a dataset. These will be discussed in Section 2.5.

2.1.2 Approaches to Text Classification

2.2 Vector Space Models *

Text documents cannot be used directly as input to a classifier, and are thus usually mapped into a vector space so that each document can be represented by a vector $\mathbf{v} \in \mathbb{R}^d$. This procedure is also known as *Document Indexing* [Sebastiani, 2002].

"Contiguity hypothesis. Documents in the same class form a contiguous region and regions of different classes do not overlap." [Manning et al., 2008, Chapter 14, p. 289]

Vector space model [Manning et al., 2008, Chapter 6.3, p. 120]

"the document 'Mary is quicker than John' is, in this view, identical to the document 'John is quicker than Mary'"[Manning et al., 2008, Chapter 6.2, p. 117]

2.2.1 N-gram Models

N-gram language models are based on co-occurrences of word or character sequences, so-called N-grams or k-shingles as they are referred to in the Data Mining literature [Leskovec et al., 2014, Chapter 3.2, p. 72]. Formally an N-gram is defined as a sequence of n items, each of which consist of n characters or words, effectively used to capture sub-sequences of text. Common choices are N-grams of size 1, 2 or 3 — called *unigrams*, *bigrams* and *trigrams* respectively — and the definition can be extended to using a window size $[w_{\min}, w_{\max}]$, employing all combinations of N-grams in this interval.

N-grams are usually used to create a vector-space model by representing each document in a dataset as a *bag-of-words* or *bag-of-N-grams* vector so that each dimension of the vector represents statistics about the corresponding N-gram. Specifically, a common way to compute the word count vectors for a document is the following:

$$\mathrm{TF}_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \tag{1}$$

Where f_{ij} is "the *frequency* (number of occurences) of a term (word) i in document j" and TF_{if} is the *term frequency*, i.e. "f_{ij} normalized by dividing it by the maximum number of occurrences of any term [...] in the same document" [Leskovec et al., 2014, Chapter 1.3.1, p. 8].

example with 2 vectors and showing what

Variants As this approach has been studied for decades there is quite an extensive amount of variants and thus hyper-parameters to tune. The most important ones will be explained in the following sections:

citation for first or review paper here?

Words vs. Characters The first choice when building an N-gram language model is to use characters or words as the atomic unit. In practically every case there are less characters than words in a dataset, but to capture expressive substrings usually larger N-gram window sizes or ranges have to be chosen, which leads to a combinatorial explosion. In case of word-based models on the other hand the maximal size of the feature space is the size of the vocabulary $\mathcal{V}$ in the case of unigrams or V^k in case of k-grams.

Stop words For creating N-gram models, so-called stop word lists are often used which are lists of frequent words that will be excluded as they do not carry much meaning [Leskovec et al., 2014, Chapter 1.3.1, p. 7]. The stop-word list used in these experiments is the standard list used for the Scikit-learn framework [Pedregosa et al., 2011] which is a list gathered by the University of Glasgow Information Retrieval group[4].

N-gram range The N-gram range, also known as window size or shingle size, refers to combinations of the atomic units of the model (words or characters) and defines an upper and lower limit for these combinations. For example a range of $[1, 1]$ specifies a unigram model, $[2, 2]$ a bigram model and $[1, 2]$ a combination of both including all unigrams and all bigrams. A larger range allows the model to capture an increasing amount of word order and thus context, but again leads to a combinatorial explosion in terms of feature space.

Vector size The vector size imposes an upper limit to the vector size and therefor the number of N-grams that can be encoded in the feature space. Commonly this simply uses the words with the highest frequency to reduce the vector size from the full length — the size of the vocabulary — to the desired size.

TF.IDF weighting A common extension to using word-counts is to weight the term frequencies by the so-called inverse document frequency, i.e. the inverse of the frequency of an term or N-gram in all documents. This method is commonly referred to as *TF.IDF* and specifically the inverse document frequency is defined as $\text{IDF}_i = \log_2(N/n_i)$, where logarithmic smoothing is applied. The TF.IDF value for a term or N-gram is then computed as $\text{TF}_{ij} \cdot \text{IDF}_i$.

[4] http://www.gla.ac.uk/schools/computing/research/researchoverview/informationretrieval/. The full stop word list can be found at http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words and in the appendix in Section **??**.

Sublinear TF scaling As [Manning et al., 2008, Chapter 6.4.1, p. 126] suggests "[it] seems unlikely that twenty occurrences of a term in a document truly carry twenty times the significance of a single occurrence". Hence a common variant is *sublinear scaling* where we down-weigh the increase in term importance by applying a logarithmic function to it, resulting in the sub-linear term frequency subTF_{ij}:

$$\text{subTF}_{ij} = \left\{ \begin{array}{ll} 1 + \log \text{TF}_{ij} & \text{TF}_{ij} > 0 \\ 0 & \text{otherwise} \end{array} \right\}$$

Normalization Often the term vectors are globally normalized using the L_1 or L_2 norm to remove the effect of statistical differences between the terms.

There are, of course, various other variants and modifications to the N-gram model, but within the scope of this thesis only the most notable ones were introduced and will be used for experiments later. For further material on this subject refer for example to [Manning et al., 2008].

mention smoothing techniques [Chen and

Shortcomings Today N-gram models are still in wide use and considered as state of the art "not because there are no better techniques, but because those better techniques are computationally much more complex, and provide just marginal improvements" [Mikolov, 2012, p. 17]. As [Mikolov, 2012] points out further "[the] most important weakness is that the number of possible n-grams increases exponentially with the length of the context, preventing these models to effectively capture longer context patterns. This is especially painful if large amounts of training data are available, as much of the patterns from the training data cannot be effectively represented by n-grams and cannot be thus discovered during training. The idea of using neural network based LMs [Language Models] is based on this observation, and tries to overcome the exponential increase of parameters by sharing parameters among similar events, no longer requiring exact match of the history H." [Mikolov, 2012, p. 17]

2.2.2 Language Models using Distributed Representations

To overcome the shortcomings of popular language models such as the ones of the N-gram model mentioned above, lots of recent work went into the study of so-called distributed language models. One branch of research that gained significant attention is the work on Neural Network based Language models (NNLMs), popularized largely through the work of T. Mikolov and his software realization of such a model dubbed *word2vec* with interest coming not only from the academic

Figure 2: Google Trends statistics on relative search interest in the term "word2vec". Retrieved on 22.05.2016.

community but also from open source community (Figure 2 shows the search relevance of the term "word2vec" in the recent years). His work builds on ideas introduced in [Bengio and Bengio, 2000] where a neural network based model was proposed for modeling high-dimensional discrete data, which was then applied to the domain of language modeling in [Bengio et al., 2003]. Following the description in this paper, the approach is as follows:

1. Associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$),
2. Express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence, and
3. Learn simultaneously the *word feature vectors* and the parameters of that *probability function*.

To achieve this, a feedforward neural network model is trained to learn these *word feature vectors* or *word embeddings*. As input a sequence of n words is given, each encoded using one-hot encoding or one-of-V encoding where the corresponding indicator vectors for each word have the size of the vocabulary V. The input word vectors are then projected linearly into a projection layer of significantly lower dimensionality D, using a global projection matrix for across all words, and concatenated, forming the input of size $D \times N$ to a hidden layer of size H. The hidden layer then feeds non-linearly into the output layer that is again of size V, modeling the probability distribution for a word given its context $P(w_t \mid w_{t-n}, \ldots, w_{t-2}, w_{t-1})$.

Simplified Continuous Models [Mikolov et al., 2013a] then introduced two simplified models, removing the hidden layer and only using a projection layer, with shared weights for all words. The Continuous Bag-of-Words Model (CBOW) model is trained to predict the current word w_t given the k words around it. Its name is due to the fact that the word order does not influence the projection as the word vectors are summed or averaged. The Continuous Skip-gram Model works the other way around, predicting the most likely k words around a given word w_t. Figure 3 illustrates both models.

These models have been shown to outperform state of the art N-gram models on various tasks (see e.g. [Bengio et al., 2003] or [Mikolov, 2012]). An interesting outcome of this research is the fact that these *word vectors* capture many interesting and often subtle semantic regularities and that these

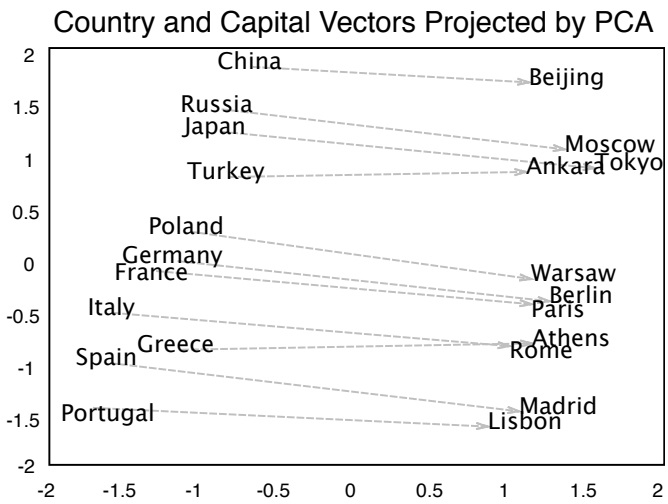
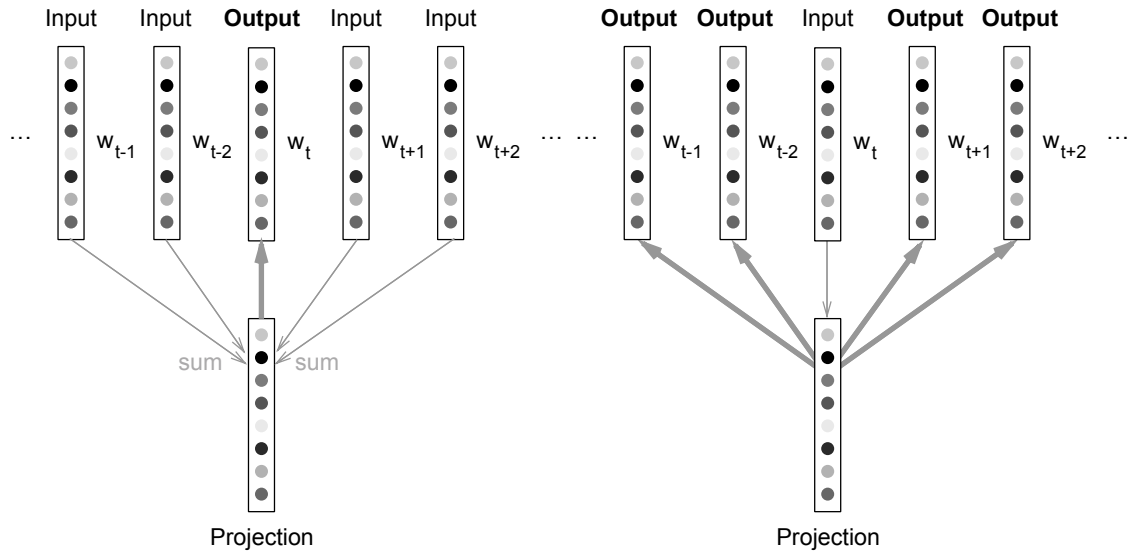


Figure 4: Two-dimensional PCA projection of the



(a) Continuous Bag-of-Words Model (b) Continuous Skip-gram Model

Figure 3: Architectures for learning continouus distributed word vectors, adapted from [Mikolov et al., 2013a]

can be exploited explicitly in an algebraic manner. When trained on an extensive dataset, one can perform calculations as $v(Paris) - v(France) + v(Germany)$ and the closest vector to the result turns out to be $v(Berlin)$ where $v(\cdot)$ denotes the *word vector* of a word. Figure 4 shows a PCA projection of Skip-gram trained vectors of countries and their capital cities.

A notable alternative to these models was developed by [Pennington et al., 2014]. In their model called *GloVe*, which stands for global vectors, they construct a vector space model with similar properties as the models introduced above, which instead relies global word-word co-occurrence counts. This method thus operates directly in the co-occurrence statistics of the corpus compared to the Neural Network based methods that "fail[...] to take advantage of the vast amount of repetition in the data" [Pennington et al., 2014].

There have been various extensions and variants to the Neural Network based language models especially, including architectures based on Recurrent Neural Networks (see [Mikolov, 2012]). Some of the most important variations will are discussed in the following section as they were evaluated in the experiments:

Hierarchical Softmax The architectures proposed in [Bengio and Bengio, 2000], [Bengio et al., 2003] and follow-up work use a *softmax* activation function at the output layer in order to obtain valid probabilities for each word to be predicted:

$$\text{softmax}(\mathbf{x}_j) = \frac{\exp(\mathbf{x}_j)}{\sum_k \exp(\mathbf{x}_k)} \tag{2}$$

Hierarchical Softmax uses a binary tree to encode the output which leads to an efficient approximation of the full softmax and speeds up training and inference. Details can be found in [Mikolov et al., 2013b].

Negative Sampling Another technique applied by [Mikolov et al., 2013b] *Negative Sampling* which is a simplified version of Noise Contrastive Estimation (NCE) introduced by [Gutmann and Hyvärinen, 2012]. Based on the insight that a good model should be able separate noise from signal, this method mixes samples from a noise distribution into the signal to be learned, in this case random words that are not in the context window, which is shown to approximately maximize the log probability of the softmax. Free parameters of this technique are the number of negative samples k per data sample and the noise distribution $P_n(w)$

Sub-sampling of Frequent Words As there the difference between frequent and infrequent words in large corpora can be huge and the frequent words often don't carry as much meaning, in [Mikolov et al., 2013b] a simple sub-sampling technique is used to counter this imbalance by discarding words with a probability computed as follows:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_f)}} \tag{3}$$

with $f(w_i)$ denoting the frequency of word w_i and t denoting a threshold. [Mikolov et al., 2013b] state that this method, while chosen heuristically, "accelerates learning and even significantly improves the accuracy of the learned vectors of the rare words".

Distributed representations for documents The models explained above are defined on words as the atomic unit. Therefore several ways have been proposed to extend these to sequences of words in order to obtain a vector space of sentences or documents. A few of these will be briefly outlined here:

Bag-of-Means The term *Bag-of-Means* refers to simply averaging over the word embedding vectors of all words in a document. However this approach "loses the word order in the same way as the standard bag-of-words models do." [Le and Mikolov, 2014]. This intuitive property was confirmed by [**?**] where the method consistently performed poorest in comparison to other approaches on a variety on tasks.

Parse Trees [Le and Mikolov, 2014] also mention a more sophisticated approach by "combining the word vectors in an order given by a parse tree of a sentence." as done in [Socher et al., 2011], with the disadvantage that this method "has been shown to work for only sentences because it relies on parsing" [Le and Mikolov, 2014].

Paragraph Vectors * In [Le and Mikolov, 2014] a different approach is shown that builds on the same idea as the original word2vec model:

2.3 Classification Algorithms for Vector Space Models

Classification Schemes As will be discussed later in Section 2.5, there are three common schemes for classification: In *binary classification* there is only a single class and for each document we decide whether or not it belongs to this class. A classic example is email spam detection where we predict if a given email is spam or not. *Multi-class classification* assumes the existence of more than one class and can be sub-categorized into *single-label classification* where the labels are mutually exclusive and and *multi-label classificaion* where they are not and thus multiple labels can be assigned to a single document at the same time.

2.3.1 Generalized Linear Models

2.3.2 Baysian Classifiers

2.3.3 Decision Trees

2.3.4 Example-Based Classifiers

2.3.5 Ensemble Methods

2.3.6 Support Vector Machines

2.3.7 Neural Networks

2.4 Sequential Text Classification

2.5 Evaluation

In this section the basics of evaluating classification models for the given problem will be laid out. First the different evaluation schemes and their advantages or disadvantages are explained in the dichotomous case where only one class is to be predicted in terms of being active or not. Then these are generalized to the multi-label case where K mutually exclusive classes are given. The last section extends this concept again towards so-called multi-label multi-output classification where several output labels can be predicted at the same time.

2.5.1 Binary Classification

In the binary case of classification we are given a single class k and a set of labelled data points $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where targets $y_i \in \{0, 1\}$ encode

whether a data point x_i belongs the class c or not. The task is then to achieve correct classification of new data points without knowing the true label via a model function or predictor $f(\cdot)$.

To evaluate such a predictor it is useful to present the results in form of a contingency table as shown in table Table 1, because it gives valuable insights about the performance of the prediction. The table shows the proportion of data points that belong to the class (RP) or not (RN) and were predicted correctly (TP) or incorrectly (FN), as well as the number of data samples that do not belong to the class (RN) and were falsely predicted to be in the class (FP) or correctly predicted to not be in the class (TN), and the same proportions for the positively (PP) and negatively (PN) predicted cases with respect to the true assignments to the data. N refers to the total amount of data points.

	Real Positives (RP)	Real Negatives (RN)
Predicted Positives (PP)	True Positives (TP)	False Positives (FP)
Predicted Negatives (PN)	False Negatives (FN)	True Negatives (TN)

Table 1: Contingency table for binary classification

Accuracy An intuitive choice towards classification is to simply ask which data points were correctly classified to belong to the class or not. In terms of the contingency table above the ratio of (TP + FP)/(N), commonly referred to as the "accuracy" of the classifier.

This choice can give a good intuition and it does capture the effectiveness on both true positives as well as true negatives, but it is strongly influenced by bias of the true and predicted class distribution (known as prevalence RP/N and label bias) as pointed out by [Powers, 2011]. For example given a population of 900 positive and 100 negative examples, a predictor that simply always chooses a positive assignment can achieve accuracy of 90% while it obviously is not a great predictor.

"There is a good reason why accuracy is not an appropriate measure for information retrieval problems. In almost all circumstances, the data is ex- tremely skewed: normally over 99.9% of the documents are in the nonrele- vant category. A system tuned to maximize accuracy can appear to perform well by simply deeming all documents nonrelevant to all queries. Even if the system is quite good, trying to label some documents as relevant will almost always lead to a high rate of false positives. However, labeling all documents as nonrelevant is completely unsatisfying to an information retrieval system user. "[Manning et al., 2008, Chapter 8.3, p. 155]

Precision, Recall and F1 Score In the field of Information Retrieval it is common practice to measure the effectiveness of a predictive system in terms of its precision and recall. The precision of such system is "the proportion of retrieved material that is actually relevant" whereas the recall measures "proportion of relevant material

actually retrieved in answer to a search request" [Rijsbergen, 1979]. Formally these two measures are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

As both, high precision and recall, are important for an robust information retrieval system they are typically combined into a single measure such as the F-measure, also referred to as F-score. The F-score is the weighted harmonic mean between precision and recall, derived from the measure of effectiveness proposed in [Rijsbergen, 1979]. The most common form is the F_1 score where precision and recall are assigned equal weight:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

The F_1 score has the advantage of its intuitive interpretability as both precision and recall are well understood measures and, analogous to recall, precision and accuracy, as it lives in the range $[0, 1]$, giving a single number that can express the effectiveness of the system in terms of percentage.

The F1 score is widely used in the field of Machine Learning and Data Mining and thus it is an important measure to consider to compare results to outcomes of prior publications by others. It is however important to point out that any version of the F-measure is a biased score as it "ignores TN which can vary freely without affecting the statistic" [Powers, 2011]. This can affect the evaluation of a classifier when the class distribution is skewed (prevalence) or the classifier develops a bias towards certain classes (label bias), motivating the use of unbiased measures in these cases, such as the ones described next.

Informedness, Markedness and Matthews Correlation Coefficient [Powers, 2011] introduces unbiased analogue measures to Recall and Precision, called "Informedness" and "Markedness" respectively. As [Powers, 2011] lays out, "Informedness quantifies how informed a predictor is for the specified condition, and specifies the probability that a prediction is informed in relation to the condition (versus chance).":

$$\begin{aligned} \text{Informedness} &= \text{Recall} + \text{Inverse Recall} - 1 \\ &= 1 - \text{Miss Rate} - \text{Fallout} \\ &= 1 - \frac{\text{FN}}{\text{RN}} - \frac{\text{FP}}{\text{RP}} \end{aligned} \tag{7}$$

Further he defines: "Markedness quantifies how marked a condition is for the specified predictor, and specifies the probability that a condition is marked by the

predictor (versus chance)."

$$\begin{aligned} \text{Informedness} &= \text{Recall} + \text{Inverse Recall} - 1 \\ &= 1 - \text{Miss Rate} - \text{Fallout} \\ &= 1 - \frac{\text{FN}}{\text{RN}} - \frac{\text{FP}}{\text{RP}} \end{aligned} \tag{8}$$

Based on Informedness and Markedness we can then see that *Matthews Correlation Coefficient* r_G, first proposed by [Matthews, 1975], is a score that balances these two measures:

$$\begin{aligned} r_G &= \pm\sqrt{\text{Informedness} \cdot \text{Markedness}} \\ &= \frac{(\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN})}{(\text{TP} + \text{FN})(\text{FP} + \text{TN})(\text{TP} + \text{FP})(\text{FN} + \text{TN})} \end{aligned} \tag{9}$$

Matthews Correlation Coefficient can thus be used as unbiased alternative to the F-measure and offers a similar ease of interpretability as it ranges from -1 to 1, the former indicating a negative correlation or adverse estimation and the latter indicating a perfect prediction, while a coefficient of 0 reflects chance.

Cross-Entropy Another common way to evaluate classifiers is the *cross-entropy* loss function:

$$\mathbb{H}(p, q) = -\sum_{n}^{N} p_n \log q_n \tag{10}$$

where p and q are discrete probability distributions. The *cross-entropy* can be derived from the *KL-divergence* as in [Murphy, 2012, Chapter 2.8.2, p. 57]:

$$\begin{aligned} \mathbb{KL}(p, q) &= \sum_{n}^{N} p_n \log \frac{p_n}{q_n} \\ &= \sum_{n}^{N} p_n \log p_n - \sum_{n}^{N} p_n \log q_n \\ &= -\mathbb{H}(p) + \mathbb{H}(p, q) \end{aligned} \tag{11}$$

where $\mathbb{H}(p)$ is the regular entropy, i.e. the lower bound on the number of bits needed to transmit the state of a random variable (as in [Shannon, 2001]), and $\mathbb{H}(p, q)$ is the cross-entropy, i.e. "the average number of bits needed to encode data coming from a source distribution p when we use model q to define our codebook" [Murphy, 2012, Chapter 2.8.2, p. 57].

In the case of binary classification we can rewrite the cross-entropy into the following error or loss function of the learned weight vector:

$$E(\mathbf{w}) = -\log p(\mathbf{T} \mid \mathbf{w}) = -\sum_{n=1}^{N} t_n \log y_n + (1 - t_n) \log(1 - y_n) \tag{12}$$

where y_n denotes $y(x_n, \mathbf{w})$, the predicted output for datapoint x_n, t_n denotes the n-th true label and $\mathbf{w}$ denotes the trained weight vector of the model, as in [Bishop, 2006, Chapter 4.3.2, p. 205]. This form is also known as the *log loss* and it is commonly used with generalized linear models and neural networks (see e.g. [Bishop, 2006, Chapter 4.3.2, p. 205] and [Alpaydin, 2014, Chapter 10.7, p. 251]).

Thus, cross-entropy is a measure which is well-motivated from a information-theoretic perspective. On the downside it does not have an upper bound which makes it hard to interpret, as compared other scores that fall into [0, 1] or similar intervals.

2.5.2 Multi-class Classification

Multi-class classification refers to a generalization of the binary case where we aim to predict for each datapoint x_i one of K labels for the classes at hand. The target space $\mathcal{Y}$ can be represented with each $y_i \in \{0, 1\}^k$, known as *one-hot encoding*, where each target is c-dimensional vector. Alternatively we can encode the targets as categorical variables $y_i \in c_1, c_2, \ldots, c_k$. The contingency table from the binary case can be extended as in table 2, which is then commonly known as *Confusion Matrix* or *Error Matrix* .

citation for Confusion Matrix?

	Real Class 1	Real Class 2	...	Real Class k
Predicted Class 1	...	...		...
Predicted Class 2	...	...		...
...				
Predicted Class k	...	...		...

Table 2: Contingency table for k classes, also referred to as Confusion Matrix

Averaging for Multi-class Recall, Precision and F1-Score By definition, Recall, Precision and thus also the F-measure are defined for the dichotomous classification case, however they can be extended towards multiple classes by averaging. Two common methods are described in [Manning et al., 2008, Chapter 13.6, p. 280]: "Macroaveraging computes a simple average over classes. Microaveraging pools per-document decisions across classes, and then computes an effectiveness measure on the pooled contingency table." It is important to note that "macroaveraging gives equal weight to each class, whereas microaveraging gives equal weight to each per-document classification decision. Because the F1 measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in microaveraging." [Manning et al., 2008, Chapter 13.6, p. 280]. Formally these averaging schemes can be defined as follows, with R denoting the Recall and P the Precision.

$$\mathrm{R}_{\text{micro}} = \frac{\sum_{k=1}^{K} \mathrm{TP}_k}{\sum_{k=1}^{K} \mathrm{TP}_k + \mathrm{FN}_k} \qquad \mathrm{R}_{\text{macro}} = \frac{\sum_{k=1}^{K} \mathrm{R}_k}{K} \tag{13}$$

$$\mathrm{P}_{\mathrm{micro}} = \frac{\sum_{k=1}^{K} \mathrm{TP}_k}{\sum_{k=1}^{K} \mathrm{TP}_k + \mathrm{FP}_k} \qquad \mathrm{P}_{\mathrm{macro}} = \frac{\sum_{k=1}^{K} \mathrm{P}_k}{K} \tag{14}$$

And respectively:

$$\mathrm{F}_{1\mathrm{micro}} = 2 \cdot \frac{\mathrm{P}_{\mathrm{micro}} \cdot \mathrm{R}_{\mathrm{micro}}}{\mathrm{P}_{\mathrm{micro}} + \mathrm{R}_{\mathrm{micro}}} \qquad \mathrm{F}_{1\mathrm{macro}} = 2 \cdot \frac{\mathrm{P}_{\mathrm{macro}} \cdot \mathrm{R}_{\mathrm{macro}}}{\mathrm{P}_{\mathrm{macro}} + \mathrm{R}_{\mathrm{macro}}} \tag{15}$$

Matthews Correlation Coefficient for K classes [Gorodkin, 2004] introduced a way to extend Matthews Correlation Coefficient to the multi-class case using a generalization of Pearson's Correlation Coefficient. The coefficient is then defined as:

$$R_k = \frac{\mathrm{COV}(X, Y)}{\sqrt{\mathrm{COV}(X, X)\ \mathrm{COV}(Y, Y)}} \tag{16}$$

Where COV is the covariance function:

$$\mathrm{COV}(X, Y) = \sum_{k=1}^{K} w_k \mathrm{COV}(X_k, Y_k) \tag{17}$$

$$= \frac{1}{K} \sum_{n=1}^{N} \sum_{k=1}^{K} (X_{nk} - \overline{X_k})(Y_{nk} - \overline{Y_k}) \tag{18}$$

Similar extensions have been proposed, such as the Confusion Entropy (CEN) as described in [Jurman and Furlanello, 2012]. The article concludes:

> Confusion Entropy [...] is probably the finest measure and it shows an extremely high level of discriminancy even between very similar confusion matrices. However, this feature is not always welcomed, because it makes the interpre- tation of its value quite harder, expecially when considering sit- uations that are naturally very similar (e.g, all the cases with MCC=0). Moreover, CEN may show erratic behaviour in the binary case.
>
> In this spirit, the Matthews Correlation Coefficient is a good compromise between reaching a reasonable discriminancy degree among different cases, and the need for the practitioner of a easily interpretable value expressing the type of misclassification associated to the chosen classifier on the given task. We showed here that there is a strong linear relation between CEN and a logarithmic function of MCC regardless of the dimension of the considered problem. Furthermore, MCC behaviour is totally consistent also for the binary case.
>
> This given, we can suggest MCC as the best off-the-shelf evaluating tool for general purpose tasks, while more subtle measures such as CEN should be reserved for specific topic where more refined discrimination is crucial.

Thus Matthews Correlation Coefficient is the preferred measure when possible.

Categorical Cross-Entropy The *cross-entropy* loss function as defined above in Section 2.5.1 extends to the multi-class case quite naturally:

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_k) = -\ln p(\mathbf{T} \mid \mathbf{w}_1, \ldots, \mathbf{w}_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk} \tag{19}$$

where $y_{nk} = y_k(\phi_n)$, and $\mathbf{T}$ is an $N \times K$ matrix of target variables with elements t_{nk} (see as in [Bishop, 2006, Chapter 4.3.4, p. 209]). This form is also referred to as *multi-class log loss* and gives an aggregated loss over all classes.

more detail?

2.5.3 Multi-label Classification *

2.6 Visualization

2.6.1 PCA

2.6.2 t-SNE

3 Exploration

3.1 Project Brief

The objective of this thesis was to find interesting and novel ways to use the various data that get generated through Sanoma's recruitment platform *Oikotie Työpaikat*. This was stated in the research plan as follows:

> Find an application of data mining / machine learning to the customer-generated data on the recruitment platform Oikotie Työpaikat which has the potential of bringing value to the user of the platform and is technically feasible in the scope of a master's thesis. Further define and investigate a research problem that is essential to this application by researching literature and previous work on similar problems trying different approaches based on the literature using the results and learnings to create an improved approach.

3.2 Approach

3.3 Understanding structure of job ads

sus

3.4 Crowdsourced Data Collection (*)

Describe data format

In order to perform supervised learning labelled data was needed for training. Together with the process of reframing of the research problem this was approached in an iterative way. First a quick prototypical tool was built to collect labels in a crowd-sourced fashion. This allowed getting more knowledge about the problem itself, especially with regards to how humans perform the task of labelling topics of text sections, and to perform first experiments of algorithmically achieving meaningful results in agreement to human behavior on this task. Then these learnings were taken into consideration when re-scoping the research problem and according to that data was collected using the micro tasking service CrowdFlower[5], leading to a quality dataset of labelled sentences from job ads.

3.4.1 Explorative Paragraph Dataset

Picture of software setup?

[5] "CrowdFlower is a data enrichment, data mining and crowdsourcing company [...]. The company's software as a service platform allows users to access an online workforce of millions of people to clean, label and enrich data." Source: https://en.wikipedia.org/wiki/CrowdFlower, Company website: https://www.crowdflower.com

To collect first data a tool was build, consisting of a Node.js[6] server using MongoDB[7] as a database and communicating via a JSON with a simple website front-end using the mustache template engine[8]. The tool is online[9] and it's source code is publicly available on GitHub[10] with it's API documentation hosted online as well[11].

The data generated by using the free-form text description of each job ad and splitting it into paragraphs as can be seen in the software package as well[12].

The goal of this prototype tool for data collection was on the one hand to acquire data in order to carry our first experiments as fast as possible, and on the other hand to gain a deeper understanding about the research problem itself by giving an open, unbiased task to the participants. In particular the question at hand was how humans label the content of the different parts of a job ad.

The exact task given to the participants was "Describe what each section is about by adding one or more tags/keywords to it". They were shown a job ad that was split into paragraphs and besides each paragraph was a text field to enter 1 or more tags.

In a first step the tool was only shown to 3 participants to get immediate feedback if the user interface had flaws and whether the task was understood. Based on this feedback the tool was improved by providing an example for the participants and then tested with a slightly larger group of 12 persons. After correcting a few minor details in the user interface a public link was then shared via social media and other channels with as many people as possible. A few days later the tool was then also shared internally within Sanoma where it was set up as a competition to tag the most possible job ads.

In total 91 job ads were tagged, resulting in 379 tagged text sections and 358 tags.

[6] [...] Node.js is an open-source, cross-platform runtime environment for developing server-side Web applications. https://nodejs.org/

[7] "MongoDB is a free and open-source cross-platform document-oriented database [...]." Source: https://en.wikipedia.org/wiki/Node.js, Website: https://www.mongodb.com

[8] "Mustache is a simple web template system." Source: https://en.wikipedia.org/wiki/Mustache_(template_system), Website: https://mustache.github.io

[9] http://thesis.cwestrup.de/jobad-tagger/

[10] https://github.com/cle-ment/thesis-tagger

[11] http://thesis.cwestrup.de/jobad-tagger/apidoc/

[12] https://github.com/cle-ment/thesis-tagger/blob/master/pre-processing.ipynb

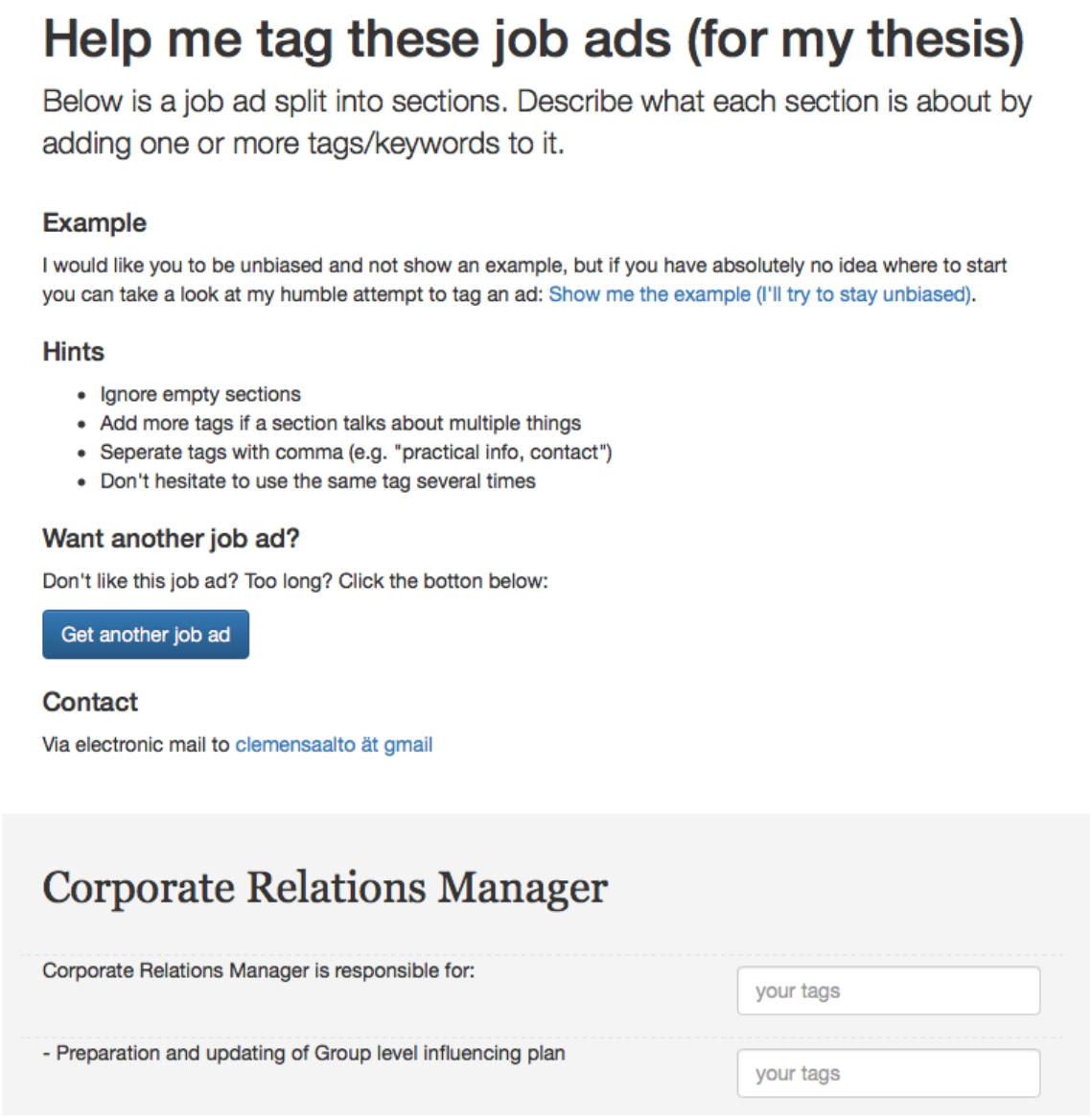


Figure 5: Screen capture of the interface of the tagging tool

Figure 6: t-SNE Embedding

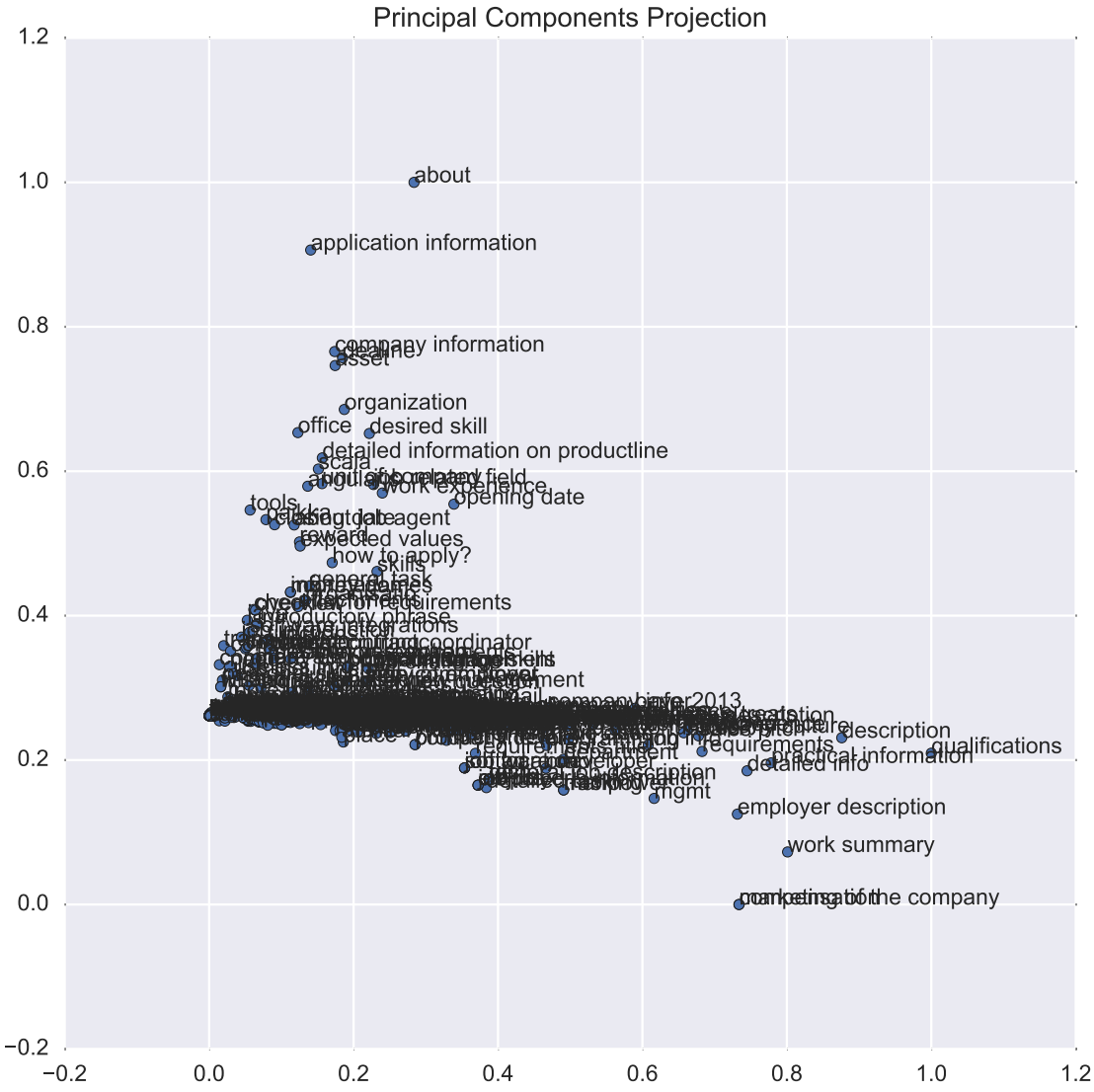
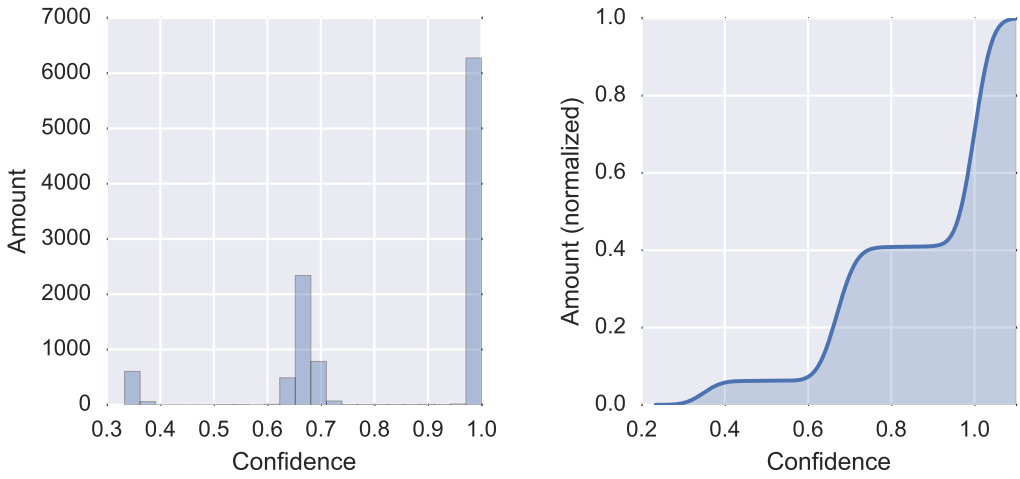


Figure 7: Principal Components Projection

4 Experimental Evaluation of Approaches to Multi-class Prediction of Semantic Categories for Text in Job Advertisements

4.1 Problem definition

4.2 Dataset



(a) Confidence (b) Cumulative Confidence

Figure 8: Amount of label judgements versus label confidence of the sentence label data collected via crowdflower

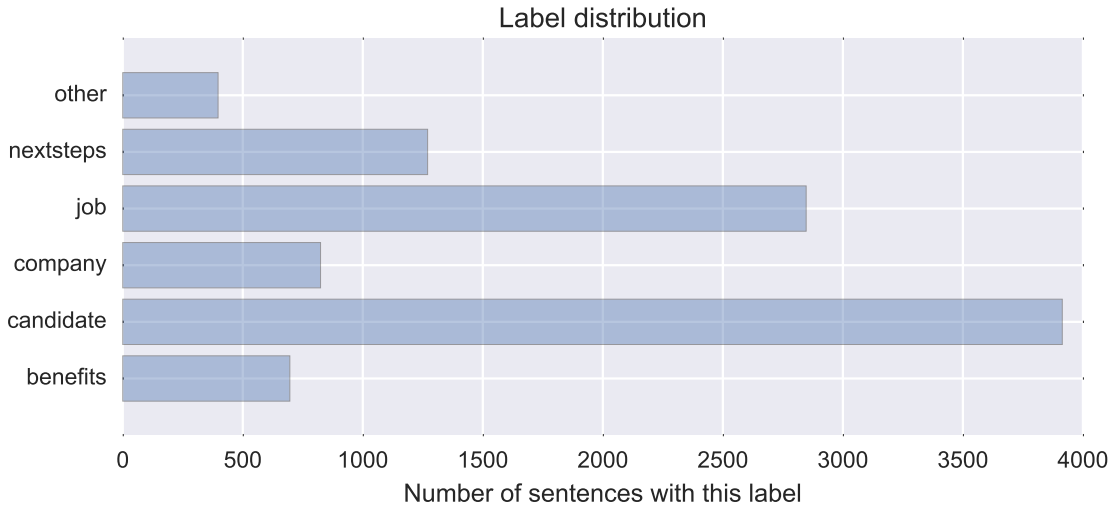


Figure 9: Distribution of labels in sentence data

4.3 Evaluation of Vector Space Models

4.3.1 Experimental Setup

As Section **??** explains, a popular way to approach text classification and other tasks in natural language processing is to build a language model by creating explicit representations of the objects or entities to be processed in a vector space. Such vectors can be used as features for a learning algorithm. Depending on the representation they can also carry further meaning, such as to encode notions of similarity of associativity between the objects.

In order to determine effective vector space representations for the task of sentence classification, a set of experiments was carried out to study and compare different approaches. Each method was studied with regards to the effect of its hyper-parameters on effectiveness when producing an input space to different classifiers, but also time and memory requirements at training and inference time are taken into account .

In order to compare the effectiveness for the sentence classification task as discussed in Sectioneach labelled document was transformed into a vector space representation using the different methods and then used for classification with a simple logistic regression classifier (). Performance was then compared with regards to Matthews Correlation Coefficient for K classes (Section 2.5.2) and the Accuracy of the classifier.

Doc2Vec model is evaluated in 2 ways (normal and trained on inferred vectors)

say why using the sentence dataset here

4.3.2 Baselines Classifiers: Uniform and Stratified Guessing

As a baseline for comparing the performance of classification two different guessing strategies were used, namely uniform and stratified guessing. Uniform guessing refers to a predictor that samples from the given classes assuming a uniform distribution whereas stratified guessing takes the label distribution in the data as the underlying probability distribution. Then both methods just sample from these distributions to produce "predictions", while ignoring the actual input data. Both, uniform and stratified guessing achieve a Matthews Correlation Coefficient score of around 0 (averaged over 1000 runs) as expected for guessing strategies (see Section **??**). On the other hand the accuracy for uniform guessing is around 0.16 which corresponds to 1/K for the K classes and around 0.26 for stratified guessing which reflects the skew of the label distribution. Figure 10 shows the confusion matrices for these baseline variants in absolute and normalized form, revealing the properties of these guessing strategies.

reference jupyter notebook here

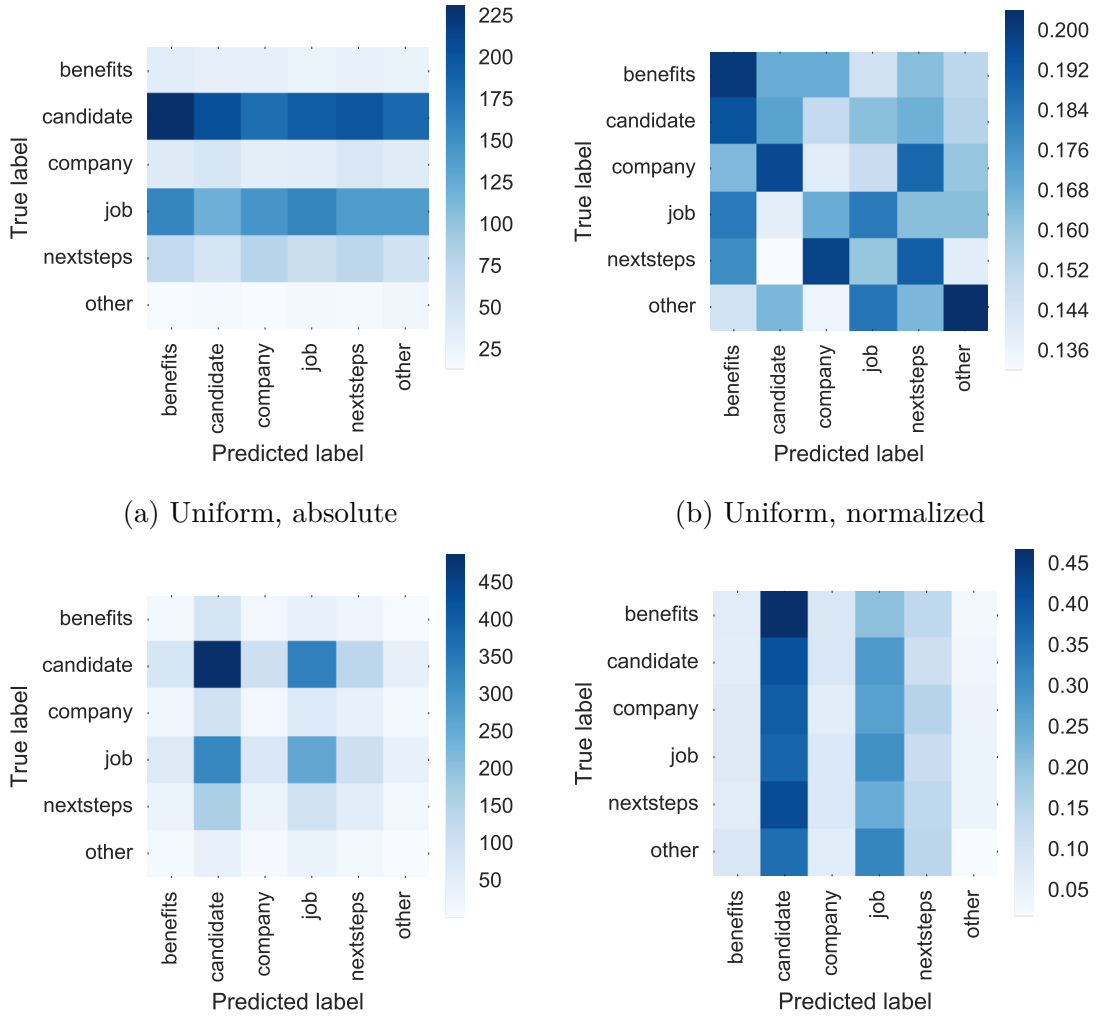
4.3.3 N-gram Language Models

The first class of language models that was investigated for the task of multi-class classification are N-gram models that were explained in Section **??**. As mentioned, in essence this type of model relies on simple statistics which makes for straightforward

actually discuss time and memory requirements

reference section here

link to



(a) Uniform, absolute

(b) Uniform, normalized

(c) Stratified, absolute

(d) Stratified, normalized

Figure 10: Confusion matrices of uniform and stratified guessing strategies.

computation but at the same time comes at cost of expressiveness, especially in terms of temporal dependencies between words.

As N-grams models come in a variety of variants the most important ones were used as hyper-parameters to the model and a grid search was carried out over a wide range of combinations over these. The specific hyper-parameter settings are listed in Table **??**. The grid search was optimized with regards to *Matthews Correlation Coefficient* (see Section 2.5.2) using 5-fold cross-validated with three standard classifiers: Logistic Regression and Naive Bayes and SVM.

The 5 best results of these exhaustive grid searches can be seen in Table 4 below.

Across all classifiers the following results on the hyper-parameters can be observed:

Why are the grid scores lower than the latter scores on the

Hyper-Parameter	N-gram Type: Words	N-gram Type: Characters
N-gram Range (Range)	[1,1], [1,2], [1,3], [2,3], [3,3]	[1,5], [1,10], [5,10], [5,15]
Stop Words	English, None	N/A
Vector Size (Size)	10, 100, 300	10, 100, 300
IDF	Yes, No	Yes, No
Norm	L1, L2, None	L1, L2, None
Sub-linear TF	Yes, No	Yes, No

Table 3: Parameter search space word and character level N-gram models

Type	Range	Stop words	Size	IDF	Norm	Sub-linear TF	MCC Score
Word	[1,1]	None	300	Yes		Yes	0.689
Word	[1,1]	None	300	Yes		No	0.687
Word	[1,1]	None	300	No		Yes	0.682
Word	[1,1]	None	300	No		No	0.682
Word	[1,1]	None	300	Yes	L2	Yes	0.68
Word	[1,1]	None	300	No		Yes	0.659
Word	[1,1]	None	300	No		No	0.656
Word	[1,2]	None	300	No		Yes	0.655
Word	[1,2]	None	300	No		No	0.655
Word	[1,3]	None	300	No		No	0.65
Word	[1,1]	None	300	Yes		Yes	0.689
Word	[1,1]	None	300	Yes		No	0.689
Word	[1,2]	None	300	Yes		Yes	0.677
Word	[1,2]	None	300	Yes		No	0.677
Word	[1,3]	None	300	Yes		Yes	0.674

Table 4: Top 5 results of grid search over hyper-parameter space as listed in Table 3 using 5-fold cross-validated Logistic Regression (top), Naive Bayes (middle) and SVM (bottom) classifiers.

Type Words as the atomic unit for N-grams consistently lead to better results. This is understandable as the search space of combinations of characters is significantly larger than the search space of known words.

Range There are slight differences to be observed between the three classifiers used, but with all three models the best performance is achieved using Unigrams. Also all of the top results across all classifiers include Unigrams in the model while extending the range towards bigrams or trigrams.

Stop Words None of the top results of the performed grid searches used stop words. This is interesting as using stop-words to remove hand-picked, highly frequent words that do not carry much meaning is common practice. It seems there is information carried within these stop words. Of course this outcome is also influenced by the particular stop-list used (see Section 2.2.1).

Size (matters) For the searched settings the largest vector dimensionality of 300 achieves the best performance. This is not surprising as higher-dimensional vectors can capture more information about N-gram occurrences. However in practice the vector size must be limited as it grows with the vocabulary — potentially at an exponential rate if N-grams other than Unigrams are used. Also very high dimensionality often leads to decreased performance in terms of generalization of the model.

IDF There is no consensus between the classifiers on whether or not to weigh the N-gram frequencies by the *inverse document frequency* (see Section 2.2.1). Thus it seems advisable to lead this parameter free for and evaluate both variants with a given classifier. For logistic regression however the performance differences are marginal and so the choice for this parameter seems somewhat arbitrary.

Norm Is seems that normalizing the vectors in most cases does not lead to any performance gains. Again this is an often recommended practice but here it does not seem to add any value to the model.

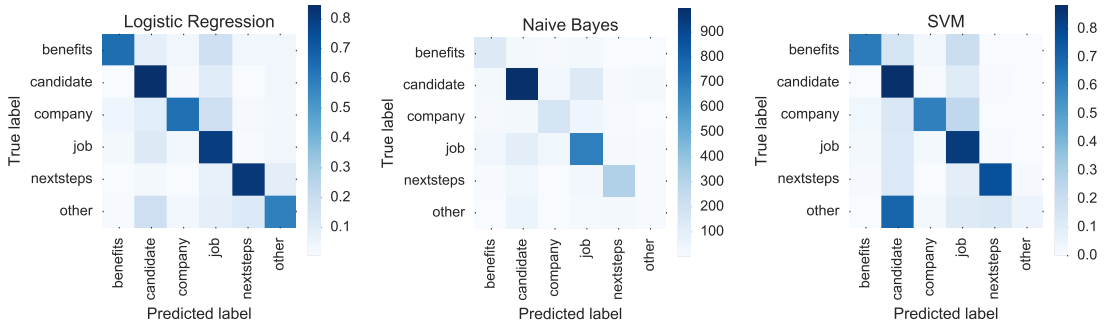
Sub-linear TF Applying sub-linear TF (see Section 2.2.1) does not seem to affect the results much and the choice of this parameter can hence be chosen almost arbitrarily as well, although here for all three classifiers applying it leads to a marginal improvement.

Table 5 shows the scores of each classifier using the best N-gram model. It is evident that here logistic regression actually performs best as it offers both, a good accuracy as well as the highest score for Matthews Correlation Coefficient.

	Training		Validation	
Classifier	Accuracy	MCC	Accuracy	MCC
Logistic Regression	0.824	0.761	0.787	0.708
Naive Bayes	0.769	0.681	0.767	0.677
SVM	0.835	0.681	0.786	0.700

Table 5: Performance of each best N-gram model with Logistic Regression and Naive Bayes on the validation data

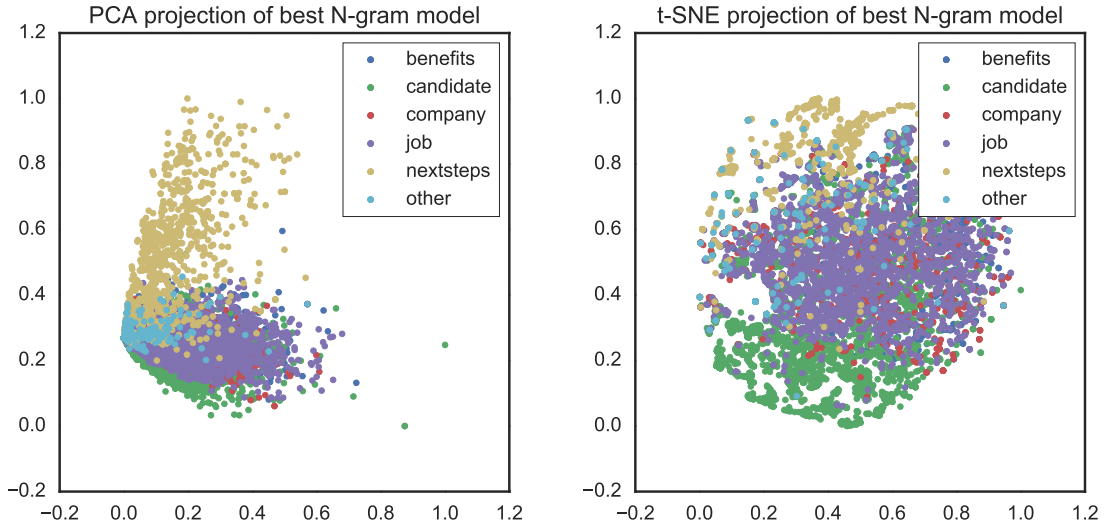
properly align visualization



(a) Logistic Regression (b) Naive Bayes (c) SVM

Figure 11: Normalized confusion matrices all three classifiers using the best N-gram model found via cross-validated grid search. Both Naive Bayes as well as SVM show label bias towards the prevalent class *candidate*.

Figure 12 shows projections of the of the constructed feature space using the best model that was optimized with Logistic Regression. This visualization shows the separability of the classes in this space. Especially the PCA projection here reveals that it is clearly possible to separate the classes until a certain point.



(a) PCA projection (b) t-SNE projection

Figure 12: Document vectors produced by the best N-gram model (optimized w.r.t. Logistic Regression) projected onto the first 2 principal components (left) and project using t-SNE projection.

4.3.4 Bag-of-Means — An Averaged Word2Vec Model

Next a Bag-of-Means model as described in Section 2.2.2 was evaluated with the same set of classifiers. The model was evaluated on the same test and training data

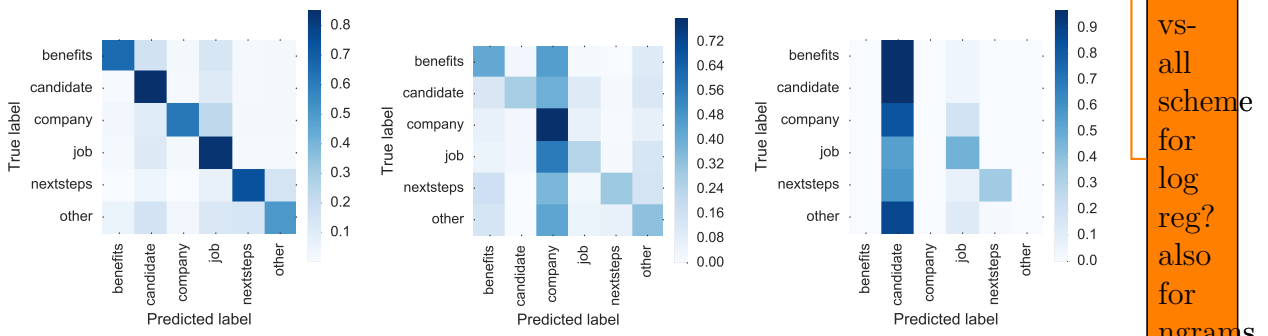
split as used for the N-gram model above. As a basis the pre-trained word-vectors from the Google News dataset[13] were used and then for each document all word vectors were average to obtain the document vector. The results can be seen in Table 6.

	Training		Validation	
Classifier	Accuracy	MCC	Accuracy	MCC
Logistic Regression	0.797	0.722	0.784	0.702
Naive Bayes	0.337	0.271	0.320	0.251
SVM	0.545	0.356	0.562	0.379

Table 6: Performance base classifiers using the Bag-of-Means model

The model performs well using Logistic Regression and is almost on par with the best N-gram model. This is surprising as performance previously reported to be rather poor as mention in Section 2.2.2. On the other hand the variance in results between the classifiers is huge and especially Naive Bayes seems to perform extremely poor. Further investigation into the use of different classifiers could shed light into these diverging results which are not observed using the N-gram models in the section above. The confusion matrices in Figure 13 reveal strong label bias in the case of Naive Bayes and SVM, although it is unclear where this stems from. Figure 14 makes clear though that there is a somewhat meaningful mapping into the feature space.

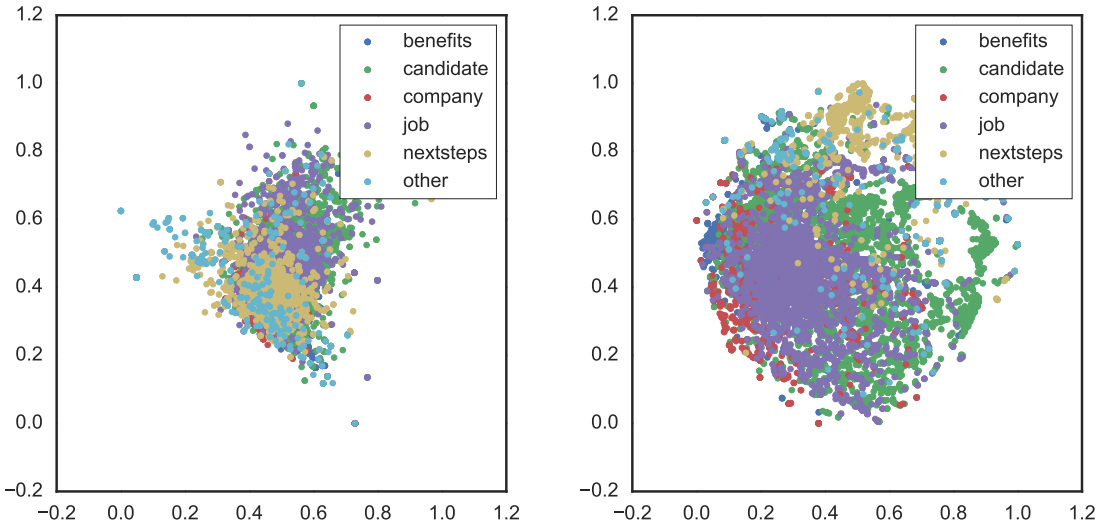
mention one-vs-all scheme for log reg? also for ngrams above



(a) Logistic Regression (b) Naive Bayes (c) SVM

Figure 13: Normalized confusion matrices of all three classifiers using the Bag-of-Means model.

[13]The dataset contains contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [Mikolov et al., 2013b]. The dataset can be obtained on the following website: `https://code.google.com/archive/p/word2vec/`



(a) PCA projection

(b) t-SNE projection

Figure 14: Document vectors produced by Bag-of-Means model (optimized w.r.t. Logistic Regression) projected onto the first 2 principal components (left) and projected using t-SNE projection. It is clear that even though the vectors are simply obtained by averaging they do indeed produce somewhat seperable manifolds.

4.3.5 Paragraph Vectors using Distributed Representations

Next a vector space model was build using the approach proposed by [Le and Mikolov, 2014] and described in more detail in Section 2.2.2. Again there are several hyper-parameters to this model that are described in Section 2.2.2 and turn out to have a huge influence on its performance as the results below indicate. As this model is computationally quite expensive a grid search as for the N-gram model above was infeasible. Thus the effect of the hyper-parameters was studied by just varying them one at a time while keeping the others fixed, using a Logistic Regression classifier with 5-fold cross-validation. The next sections will briefly outline the results of these tests:[14]

Vector Size As was to expect the vector size of the model correlates with the performance. Again the highest chosen dimensionality was 300 which yielded the best results with a Matthews Correlation Coefficient of 0.53, however the difference to a 100-dimensional model was marginal with 1% absolute improvement. Surprisingly even a 10-dimensional vector space model is capable of achieving almost best results with a difference of only 2% to the 300-dimensional model. Even a 2-dimensional model could achieve a MCC score of 14%.

[14] All tables in this section will use the following abbreviations: *type*: Model Type, i.e. PV-DM vs. PV-DBOW; *size*: Vector Size; *window*: Window Size; *negative*: Negative Sampling value k; *hs*: Hierarchical Softmax used; *sample*: Frequent word sub-sampling threshold; *MCC*: Matthews Correlation Coefficient

Vector Size	MCC Training		MCC Test	
	Trained	Inferred	Trained	Inferred
2	0.324	0.265	0.189	0.257
10	0.467	0.411	0.319	**0.404**
100	0.510	0.399	0.357	0.400
300	0.572	0.362	0.344	0.398

Table 7: Matthews Correlation Coefficient with varying vector size.

Frequent Word Sub-Sampling Frequent word sub-sampling can boost performance quite much, but again choosing the right value for this hyper-parameter is key. The training behavior with different sampling thresholds differs quite much. Figure 15 shows the training with different values with 100 passes over the dataset. A good value seems to be 10^{-5} which achieves an MCC score of 0.697 and is on-par with the best N-gram model. Interestingly not using sub-sampling in this setup seemed to be overfitting as the score decreases quite drastically with more training passes. A similar effect is observed with a higher threshold of 10^{-4} but much less strong. Choosing a lower threshold of 10^{-6} leads to very poor performance with an MCC score of only 0.07.

Threshold	Train	Test	Train (inferred)	Test (inferred)
0	0.531	0.340	0.364	0.380
1e-4	0.628	0.3151	0.372	0.401
1e-5	0.608	0.180	0.248	0.250
1e-6	0.583	0.107	0.156	0.128

Table 8: Matthews Correlation Coefficient with varying frequent word sub-sampling threshold.

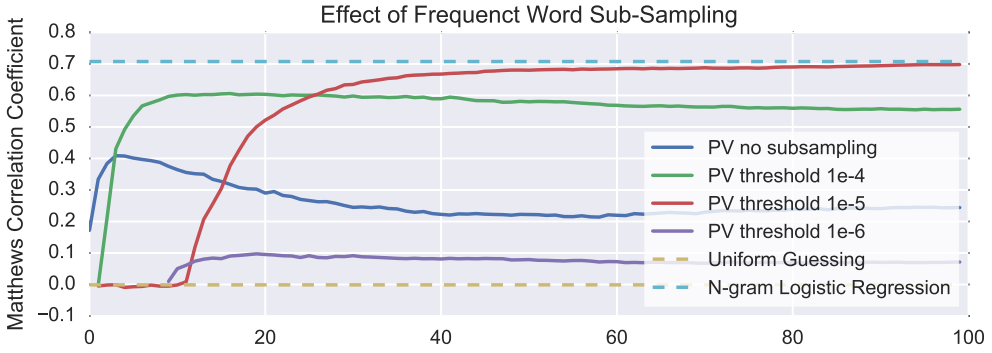


Figure 15: Training of document vectors with different sub-sampling thresholds.

Hierarchical Softmax Using hierarchical softmax increased the performance, leading to a 12% absolute difference in terms of MCC score. This result is counter-intuitive as using the hierarchical softmax should as an approximation be less performant. However it might simply mitigate overfitting of the model.

type	size	window	negative	hs	sample	MCC
PV-DM	100	10	3	0	0	0.398
PV-DM	100	10	3	1	0	0.520

Table 9: Matthews Correlation Coefficient with and without using hierarchical softmax.

Negative Sampling Negative Sampling generally increased the performance of the model and smaller values actually worked best out of the tested settings from 0 to 6. Choosing the number of negative samples to be 2 resulted in the best performance, but the absolute difference in performance was only about 6% of achieved MMC score.

type	size	window	negative	hs	sample	MCC
PV-DM	100	10	0	1	0	0.530
PV-DM	100	10	1	1	0	0.541
PV-DM	100	10	2	1	0	0.536
PV-DM	100	10	3	1	0	0.524
PV-DM	100	10	4	1	0	0.516
PV-DM	100	10	5	1	0	0.498
PV-DM	100	10	6	1	0	0.482

Table 10: Matthews Correlation Coefficient with varying negative sampling value.

Window Size Window sizes of 5, 10 and 15 were experimented with which increase or decrease the width of context the model is trained on. Here a window size of 10 showed best results. It is safe to assume that increasing the window size much further does not lead to any improvement in the model as the correlation with the word should become weaker the farther we move away from it in a document or text.

PV-DBOW versus PM-DV Both models for paragraph vectors proposed in [Le and Mikolov, 2
were tried, namely Distributed Bag of Words version of Paragraph Vector (PV-DBOW) and Distributed Memory version of Paragraph Vec- tor (PV-DM). In these tests the DBOW model achieves significantly better results with an MCC that is 14%

type	size	window	negative	hs	sample	MCC
PV-DM	100	5	3	1	0	0.508
PV-DM	100	10	3	1	0	0.523
PV-DM	100	15	3	1	0	0.510

Table 11: Matthews Correlation Coefficient with varying window size.

than the PV-DM model in absolute terms. This is in contrast with the results in the aforementioned paper, where the authors state that "PV-DM is consistently better than PV-DBOW." [Le and Mikolov, 2014].

type	size	window	negative	hs	sample	MCC
PV-DM	100	10	3	1	0	0.521
PV-BBOW	100	10	3	1	0	0.667

Table 12: Matthews Correlation Coefficient using the two models proposed in [Le and Mikolov, 2014].

Evaluating the best hyper-parameter setting Taking the learnings about the effects of the different hyper-parameters to the performance of the model a subset of models were tested in search of the best hyper-parameter selection. The results of these experiments can be seen in Table 13.

A few interesting observations can be made here. First, as indicated before, the model is highly sensitive to the settings of the hyper-parameters. Secondly we can see that the hyper-parameters interact quite strongly in some combinations. This leads to a different behavior in performance for some of the hyper-parameters than identified in the above sections, depending on what the other hyper-parameters settings are.

For instance using hierarchical softmax decreases performance when setting all other parameters to individually optimal settings, as opposed to in the previous experiment above.

write a bit more here

4.3.6 Paragraph Vectors using pre-initialized weights *

In another experiment the weight matrix for the words was initialized with pre-trained weights from from the Google News dataset.

4.3.7 Paragraph Vectors using context sentences *

This section in further research? Because

type	size	window	negative	hs	sample	MCC
PV-DM	300	10	3	0	1e-5	0.707
PV-BBOW	300	10	3	0	1e-5	0.724
PV-BBOW	300	10	3	1	1e-5	0.665

Table 13: Matthews Correlation Coefficient of different models when trying to find the best hyper-parameter setting.

4.3.8 Inversion of Distributed Language Representations (??)

4.3.9 Discussion

4.4 Evaluation of Classification Algorithms for Vector Space Models

4.4.1 Experimental Setup

4.4.2 Logistic Regression

4.4.3 Decision Tree

4.4.4 Naive Bayes

4.4.5 SVM

4.4.6 KNN

4.4.7 Random Forest

4.4.8 Neural Networks

4.4.9 Convolutional Neural Networks

4.4.10 Discussion

4.5 Evaluation of Sequential Text Classification

4.5.1 Experimental Setup

4.5.2 Character-based LSTM

4.5.3 Character-based Multi-task LSTM

4.5.4 Discussion

4.6 Results and Discussion

if this is described here it has to go into background as well

5 Discussion and Conclusions

5.1 Discussion of Experimental Results

5.2 Conclusions

- As in many areas of machine learning much work has been going into feature engineering but it seems that feature learning, while much more computationally expensive, surpasses the potential of engineered feature representations. Deep learning and meta-learning are mature enough to make up for the gap that has been there for years: To achieve performance that is good enough to make an algorithmic system usable in production, huge amounts of research and engineering went into feature engineering and finally the performance of these methods can be matched and even surpassed by automated methods or learning features. (link here NG's transfer learning work, also Schmidhubers work of meta-learning and on function prediction etc)
- There is more need to understand the representations of such feature learning systems though, statistics are quite easy to understand but weights of a neural network don't tell much. There is however potential for learning "better statistics" ourselves, e.g. how to efficiently learn a language (by looking at explicit indermediate representations of the states of a NN)
-

5.3 Contributions

- compare n-gram and doc2vec (?)
-

5.4 Proposal for Future Research

- how well do word2vec and comparable methods generalize: e.g. initialize a text corpus with word vectors from a bigger corpus (Google News), then train an RNN to predict the next word vector using the small corpus but use the bigger corpus to validate and see if words in bigger corpus can be inferred
- trajectory based algorithms (word trajectory through space for a sentence)
- Compare with standard benchmarks (TREC etc)
- Meta- / Transfer-learning: OCR with simultaneous LM learning (e.g. predict next character)
- Try on Finnish data
- Try longer parts again maybe with better seperation (paragraphs). Doc2vec gets way better accuracy when documents are longer

Trajectory -Based Algorithms on Text As shown in [Mikolov et al., 2013c] and related work that was outlined in Sectionvector space models for text can capture very subtle semantic relationships between words by their location in the vector space. When

reference here

5.5 Learnings

- focusing on both, building a working system (engineering) and exploring new directions (science), is hard
- problem framing is hard

References

[Alpaydin, 2014] Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

[Bengio and Bengio, 2000] Bengio, S. and Bengio, Y. (2000). Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3):550–557.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., and Vincent, P. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Chen and Goodman, 1996] Chen, S. F. and Goodman, J. (1996). An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

[Gorodkin, 2004] Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry*, 28(5–6):367–374.

[Gutmann and Hyvärinen, 2012] Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *J. Mach. Learn. Res.*, 13(1):307–361.

[Jurman and Furlanello, 2012] Jurman, G. and Furlanello, C. (2012). A unifying view for performance measures in multi-class prediction. *PLoS ONE*, 7(8):e41882. arXiv: 1008.2908.

[Le and Mikolov, 2014] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

[Leskovec et al., 2014] Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press.

[Lodhi et al., 2002] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text Classification Using String Kernels. *J. Mach. Learn. Res.*, 2:419–444.

[Mahoney, 1999] Mahoney, M. V. (1999). Text compression as a test for artificial intelligence. In *AAAI/IAAI*, page 970.

[Manning et al., 2008] Manning, C. D., Raghavan, P., Schütze, H., and others (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

[Matthews, 1975] Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451.

[Mikolov, 2012] Mikolov, T. (2012). *Statistical Language Models Based on Neural Networks*. PhD thesis, Ph. D. thesis, Brno University of Technology.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. arXiv: 1301.3781.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[Mikolov et al., 2013c] Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL*, pages 746–751.

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

[Powers, 2011] Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

[Rijsbergen, 1979] Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, London ; Boston, 2nd edition edition.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Comput. Surv.*, 34(1):1–47.

[Shannon, 2001] Shannon, C. E. (2001). A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55.

[Socher et al., 2011] Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

[Turing, 1950] Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, 59(236):433–460.