From a fragmented process towards integrated end-to-end learning: An explorative study of the evolution of text classification approaches towards Deep Learning

Clemens Westrup

School of Science

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

Espoo 16.1.2015

Thesis supervisor:

Michael Mathioudakis, Ph.D.

Thesis advisor:

Prof. Aristides Gionis

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Clemens Westrup

Title: From a fragmented process towards integrated end-to-end learning: An explorative study of the evolution of text classification approaches towards Deep Learning

Date: 16.1.2015 Language: English Number of pages: 6+40

Department of Information and Computer Science

Professorship: Machine Learning, Data Mining, and Probabilistic Modeling

Supervisor: Michael Mathioudakis, Ph.D.

Advisor: Prof. Aristides Gionis

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 3.4.1

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*

1.1 Corporate Partner

This thesis was done in close collaboration with the Helsinki-based company *Sanoma*. The company describes itself as follows[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.

1.2 Need Statement

Today's media and education, 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 and the wide-spread use of social media challenges traditional ways we access information.

some numbers here?

source, e.g. http://www of-the-news-media-2015/

example, e.g. fb usage compared to newspaper

Similarly, with the rise of Massive open online course, so-called MOOCs, traditional learning settings are challenged and increase the need for advanced techniques for data processing and analysis, 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.

1.3 Problem Statement *

what and why

1.4 Motivation*

1.5 Research objectives *

1.6 Related work *-

Algorithmic *text categorization* (TC — also known as *text classification*) into a fixed set of categories has been of a topic of growing interest during the last decades, boosted by the increasingly vast amounts of data available today. The applications are various, from document filtering, automated metadata generation such as language

[1] Source: http://www.sanoma.com/en/who-we-are, visited 06.06.2016

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

1.7 Structure of the thesis*

section on transfer learning and feature learning

text classification

Multitask learning

explicit vs implicit feature representation

2 Context *

2.1 Background *

2.2 Corporate Partner

This thesis was done in close and inspiring collaboration with the Helsinki-based company *Sanoma*. The company describes itself as follows[2]:

> 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.

2.3 Need Statement

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

some numbers here?

Usage of digital media as a source of information has long surpassed print media and the wide-spread use of social media challenges traditional ways we access information.

source, e.g. http://www of-the-news-media-2015/

example, e.g. fb usage compared to newspaper

Similarly, with the rise of Massive open online course, so-called MOOCs, traditional learning settings are challenged and increase the need for advanced techniques for data processing and analysis, 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.

2.4 Problem Statement *

2.5 Research objectives *

2.6 Related work *-

Algorithmic *text categorization* (TC — also known as *text classification*) into a fixed set of categories has been of a topic of growing interest during the last decades, boosted by the increasingly vast amounts of data available today. The applications are various, from document filtering, automated metadata generation such as language classification to automatic email labeling, spam identification and sentiment detection, amongst others.

[2]Source: http://www.sanoma.com/en/who-we-are, visited 06.06.2016

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

3 Background: Text Classification

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

3.1 Problem Formalism

Text classification, also known as text categorization, is the task to assign classes from a set $\mathcal{C}$ to a set of documents $\mathcal{D}$. In other words we are trying to approximate the true mapping between documents and the true classes for each document via a model function $\Phi : \mathcal{D} \rightarrow \mathcal{D} \times \mathcal{C}$. We usually assume that no additional knowledge such as metadata is available besides the text input itself. As the definition of this task is very broad additional constraints or assumptions are often imposed.

3.2 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]

3.2.1 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 $[w_{\text{min}}, w_{\text{max}}]$, 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:

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

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 they encode?

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 [3].

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

[3] `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 **??**.

$\mathrm{IDF}_i = \log_2(N/n_i)$, where logarithmic smoothing is applied. The TF.IDF value for a term or N-gram is then computed as $\mathrm{TF}_{ij} \cdot \mathrm{IDF}_i$.

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}:

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

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]

3.2.2 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.

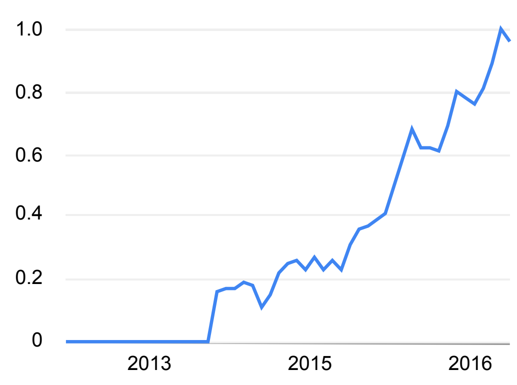


Figure 1: Google Trends statistics on relative search interest in the term "word2vec". Retrieved on 22.05.2016.

Mikolov and his software realization of such a model dubbed *word2vec* with interest coming not only from the academic community but also from open source community (Figure 1 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 $\mathbb{R}^m$),

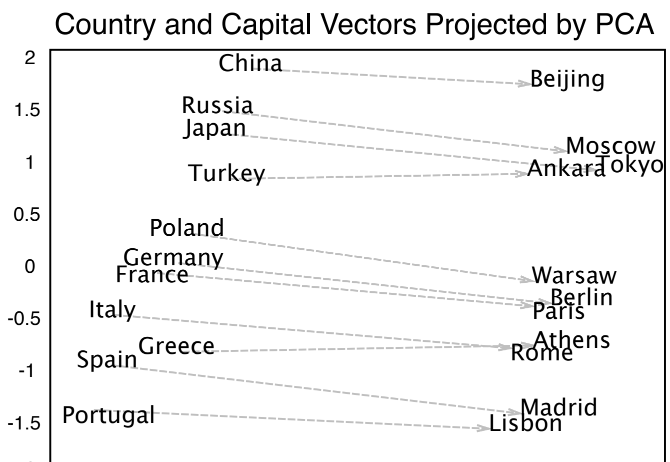
2. Express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence, and

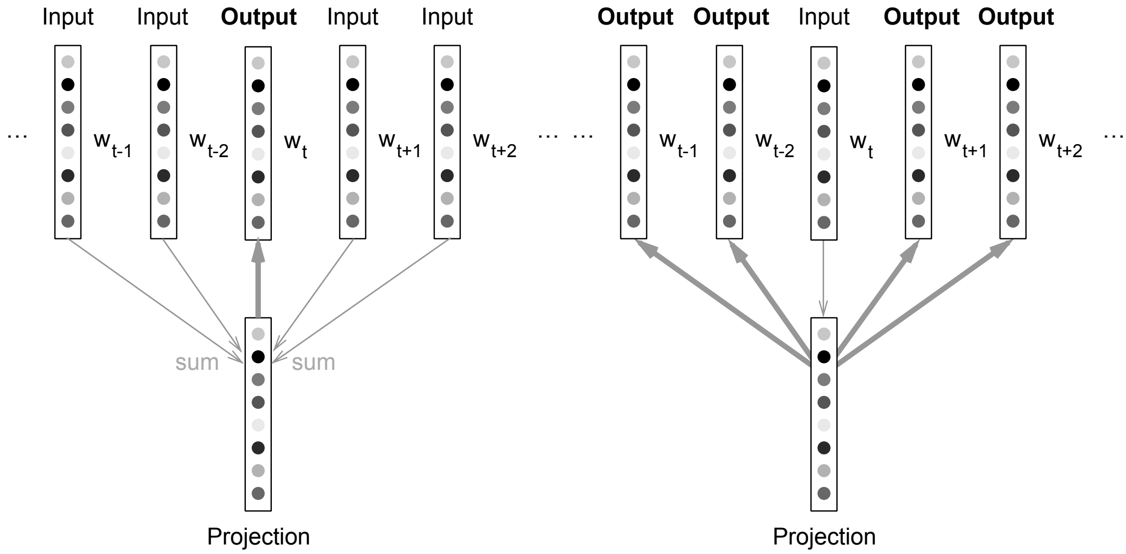
3. Learn simultaneously the *word feature vectors* and the parameters of that *probability function*.

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 2 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





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

Figure 2: Architectures for learning continouus distributed word vectors, adapted from [Mikolov et al., 2013a]

research is the fact that these *word vectors* capture many interesting and often subtle semantic regularities and that these 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 3 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:

3.2.3 Text as a Sequential Signal

3.3 Classification Approaches *

Classification Schemes As will be discussed later in Section 3.4, 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.

3.3.1 Generalized Linear Models

3.3.2 Baysian Classifiers

3.3.3 Decision Trees

3.3.4 Example-Based Classifiers

3.3.5 Ensemble Methods

3.3.6 Support Vector Machines

3.3.7 Neural Networks

3.4 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.

3.4.1 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.

3.4.2 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.

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

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

And respectively:

$$\mathrm{F_{1micro}} = 2 \cdot \frac{\mathrm{P_{micro}} \cdot \mathrm{R_{micro}}}{\mathrm{P_{micro}} + \mathrm{R_{micro}}} \qquad \mathrm{F_{1macro}} = 2 \cdot \frac{\mathrm{P_{macro}} \cdot \mathrm{R_{macro}}}{\mathrm{P_{macro}} + \mathrm{R_{macro}}} \tag{15}$$

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 3.4.1 extends to the multi-class case quite naturally:

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

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 de-tail?

3.4.3 Multi-label Classification *

4 Exploration (*)

4.1 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 microtasking service crowdflower [cro, 2016], leading to a quality dataset of labelled sentences from job ads.

4.1.1 Explorative Paragraph Dataset

Picture of software setup?

To collect first data a tool was build, consisting of a Node.js [nod, 2016] server using MongoDB[mon, 2016] as a database and communicating via a JSON with a simplistic website front-end using the mustache template engine [mus, 2016]. The tool is online[4] and it's source code is publicly available on GitHub[5] with it's API documentation hosted online as well[6].

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[7].

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.

[4] http://thesis.cwestrup.de/jobad-tagger/
[5] https://github.com/cle-ment/thesis-tagger
[6] http://thesis.cwestrup.de/jobad-tagger/apidoc/
[7] https://github.com/cle-ment/thesis-tagger/blob/master/pre-processing.ipynb

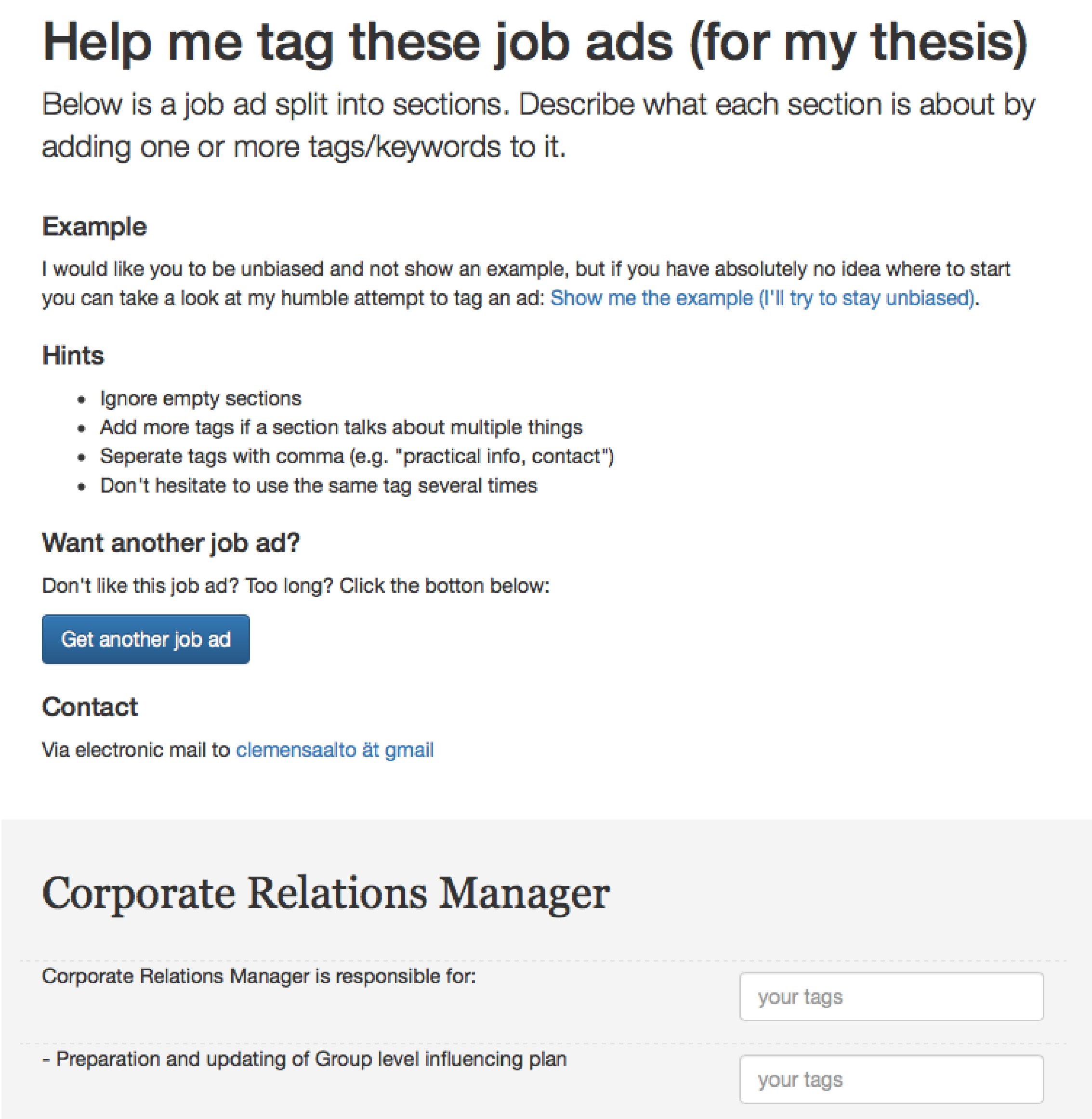


Figure 4: Screen capture of the interface of the tagging tool

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

Describe data: Different characteristics

show distribution?

show em-

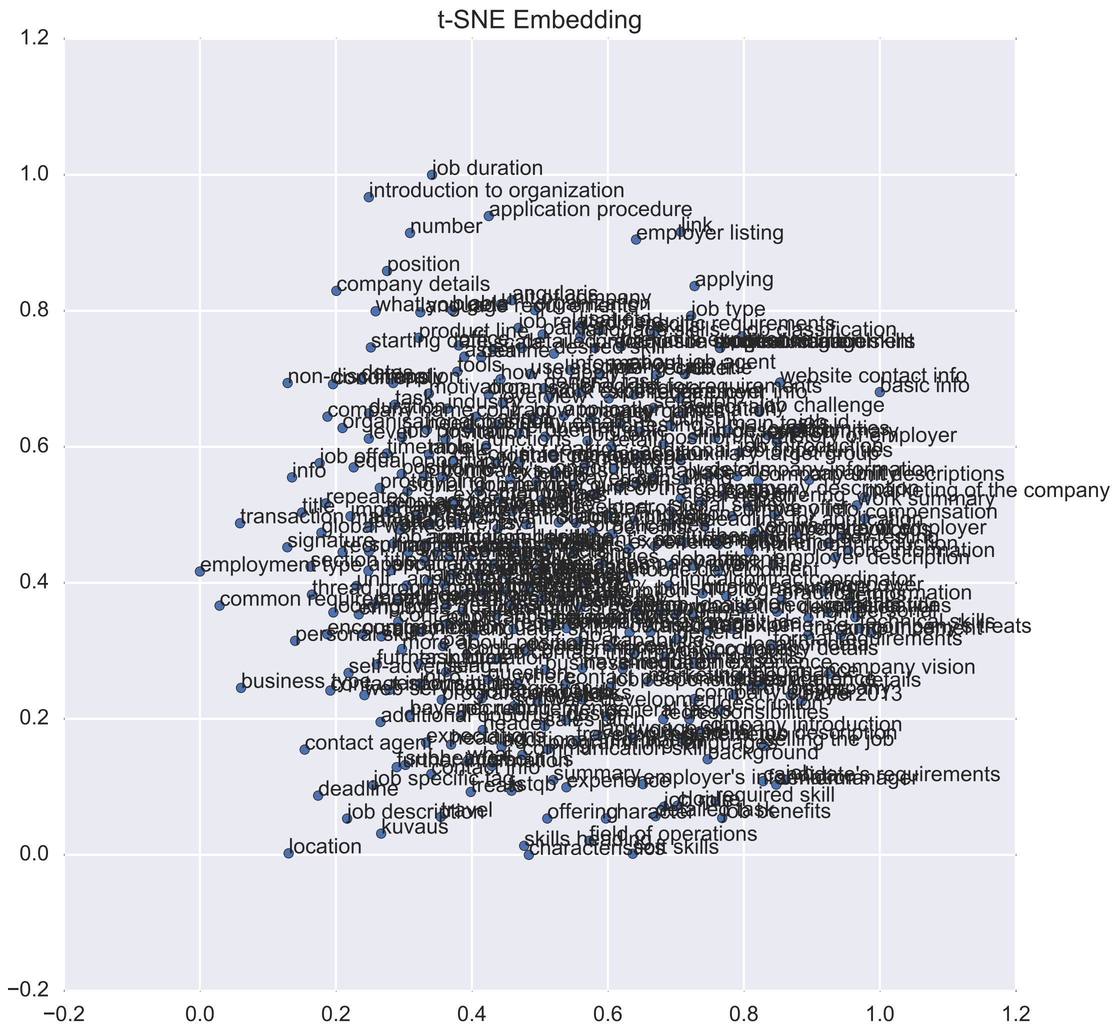


Figure 5: t-SNE Embedding

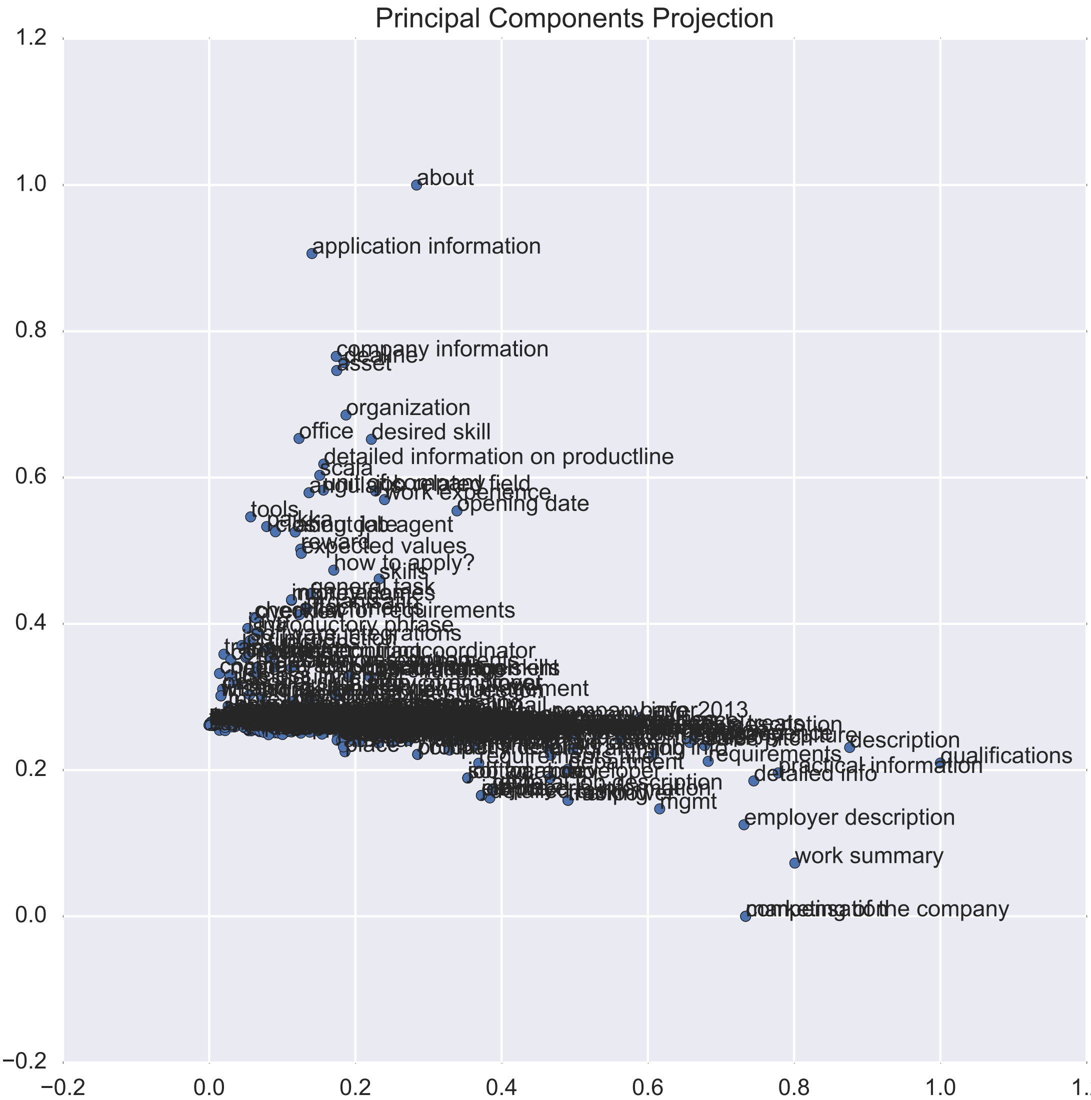
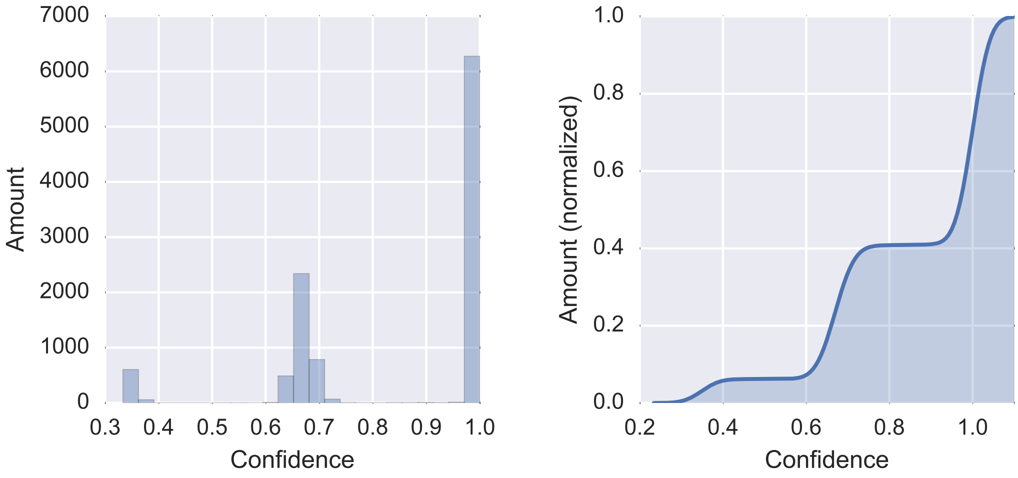


Figure 6: Principal Components Projection

5 The Meat

5.1 Problem definition

5.2 Data Collection



(a) Confidence (b) Cumulative Confidence

Figure 7: Amount of label judgements versus label confidence of the sentence label data collected via crowdflower

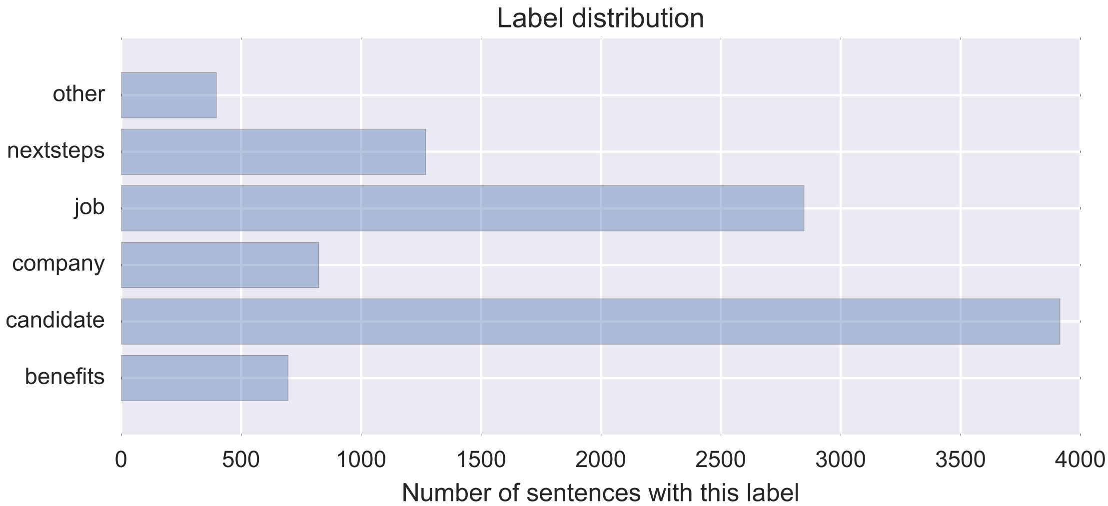


Figure 8: Distribution of labels in sentence data

5.3 Evaluation of Vector Space Models

say why using the sentence dataset here

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 .

actually discuss time and memory requirements

In order to compare the effectiveness for the sentence classification task as discussed in ?each 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 3.4.2) and the Accuracy of the classifier.

reference section here

link to logistic regression classifier explanation here

link accuracy?

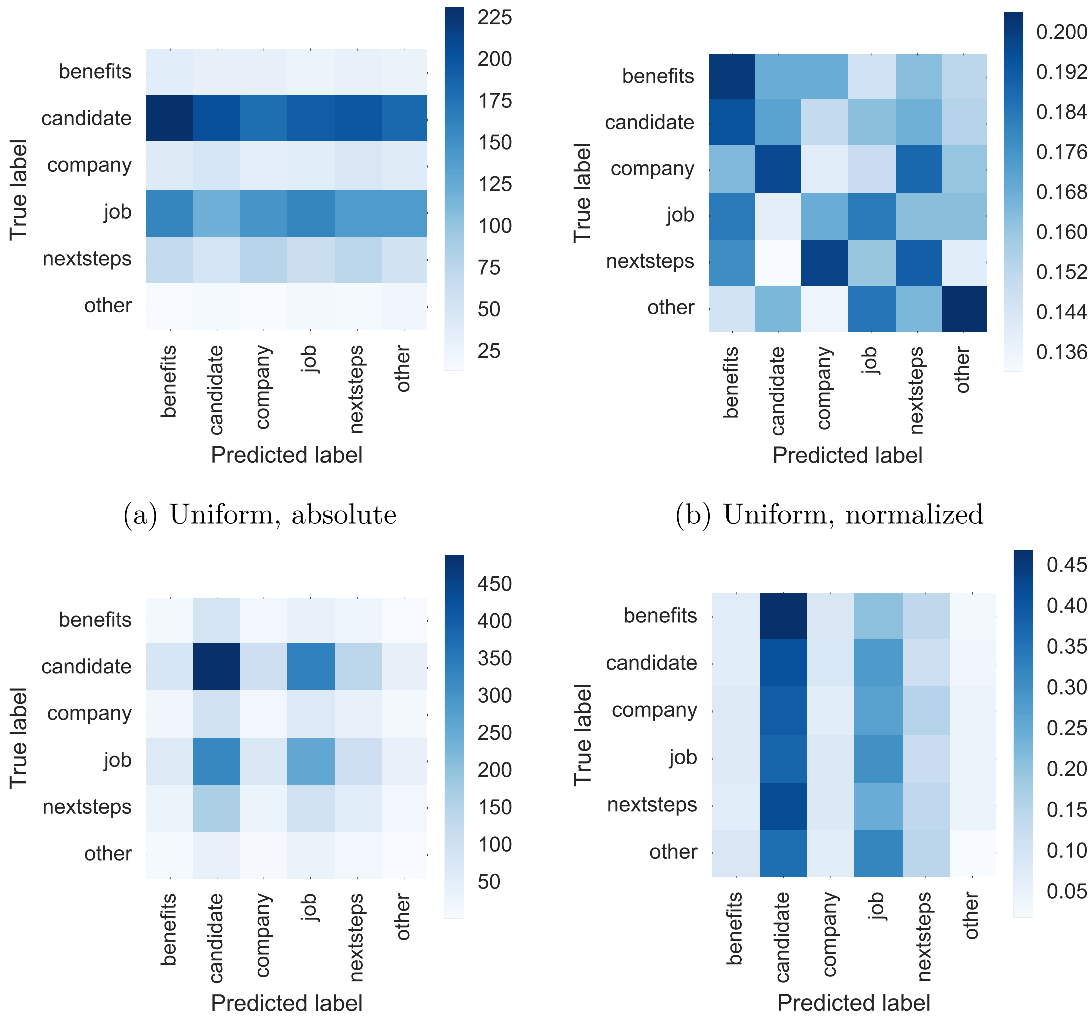
5.3.1 Baselines Classifiers: Uniform and Stratified Guessing

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 9 shows the confusion matrices for these baseline variants in absolute and normalized form, revealing the properties of these guessing strategies.

5.3.2 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 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*



(a) Uniform, absolute

(b) Uniform, normalized

(c) Stratified, absolute

(d) Stratified, normalized

Figure 9: Confusion matrices of uniform and stratified guessing strategies.

Correlation Coefficient (see Section 3.4.2) 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:

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

Why are the grid scores lower than the latter scores on the train/test split? Because they're aver-

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.

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 3.2.1).

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 3.2.1). 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 3.2.1) 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.

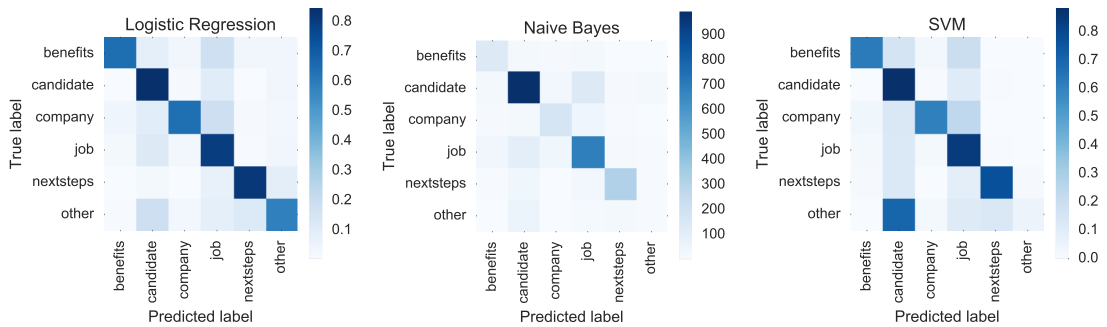
Table 5 shows the scores of each classifier using the best N-gram model. It is evident that here logistic regression actually performs best as it offers both, a good accuracy as well as the highest score for Matthews Correlation Coefficient.

	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

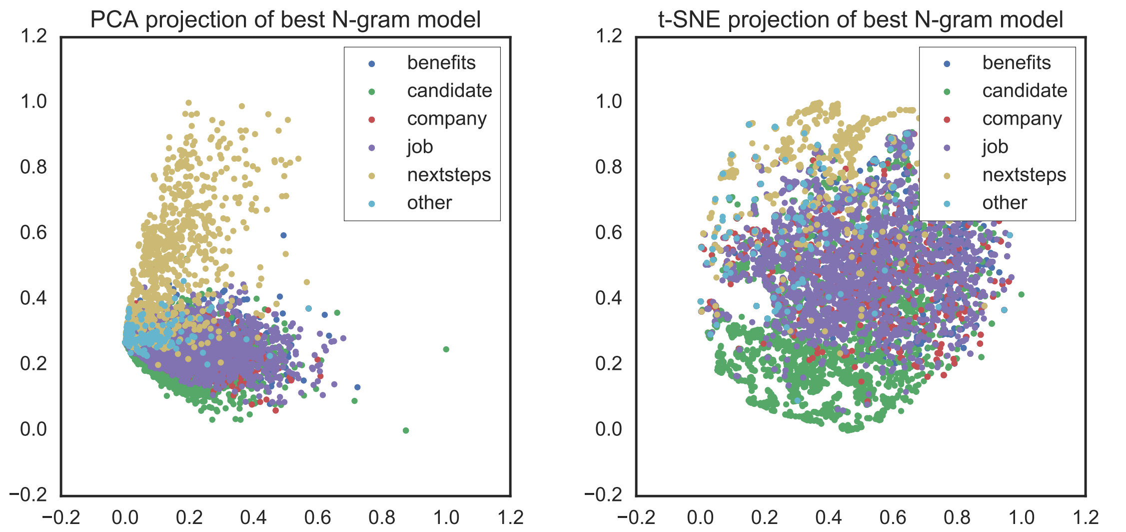
properly align visualization

Figure 11 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) Logistic Regression (b) Naive Bayes (c) SVM

Figure 10: 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*.



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

Figure 11: 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.

5.3.3 Bag-of-Means - An Averaged Word2Vec Model

Next a Bag-of-Means model as described in Section 3.2.2 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[8] were used and then for each document all word vectors

[8]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/`

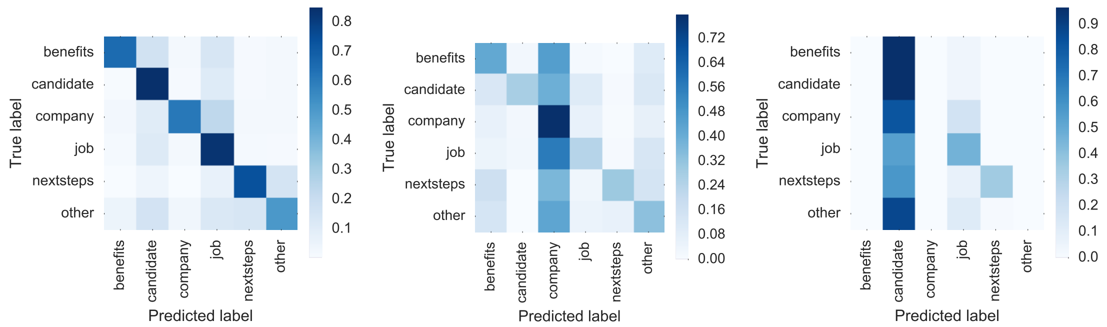
were average to obtain the document vector. The results can be seen in Table 6.

	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 3.2.2. 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 12 reveal strong label bias in the case of Naive Bayes and SVM, although it is unclear where this stems from. Figure 13 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

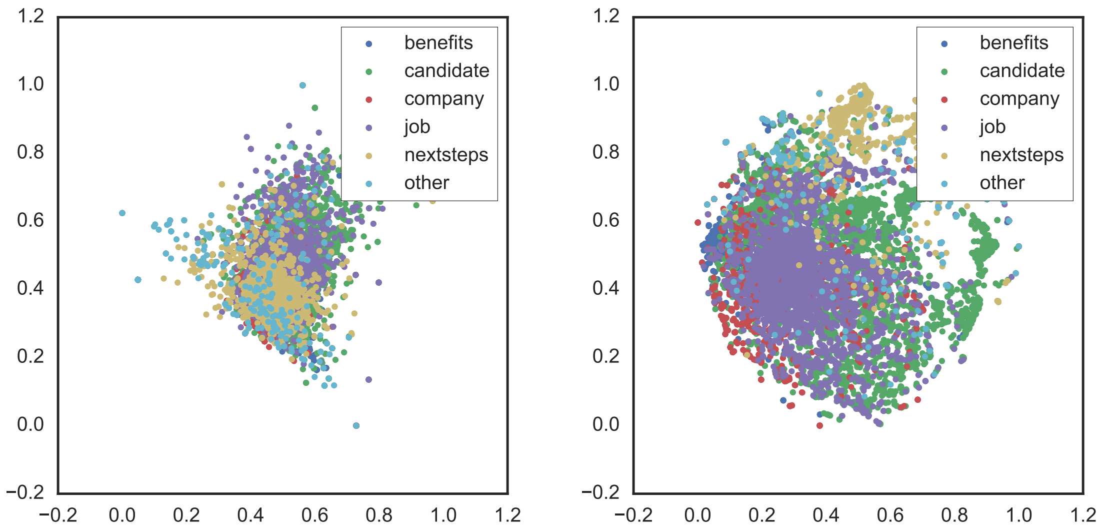


(a) Logistic Regression (b) Naive Bayes (c) SVM

Figure 12: Normalized confusion matrices of all three classifiers using the Bag-of-Means model.

5.3.4 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 3.2.2. Again there are several hyper-parameters to this model that are described in Section 3.2.2 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



(a) PCA projection

(b) t-SNE projection

Figure 13: 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.

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:[9]

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%.

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 14 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

[9] 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

type	size	window	negative	hs	sample	MCC
PV-DM	2	10	3	1	0	0.536
PV-DM	10	10	3	1	0	0.522
PV-DM	100	10	3	1	0	0.514
PV-DM	300	10	3	1	0	0.144

Table 7: Matthews Correlation Coefficient with varying vector size.

strong. Choosing a lower threshold of 10^{-6} leads to very poor performance with an MCC score of only 0.07.

type	size	window	negative	hs	sample	MCC
PV-DM	300	10	3	1	0	0.244
PV-DM	300	10	3	1	1e-4	0.556
PV-DM	300	10	3	1	1e-5	0.698
PV-DM	300	10	3	1	1e-6	0.071

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

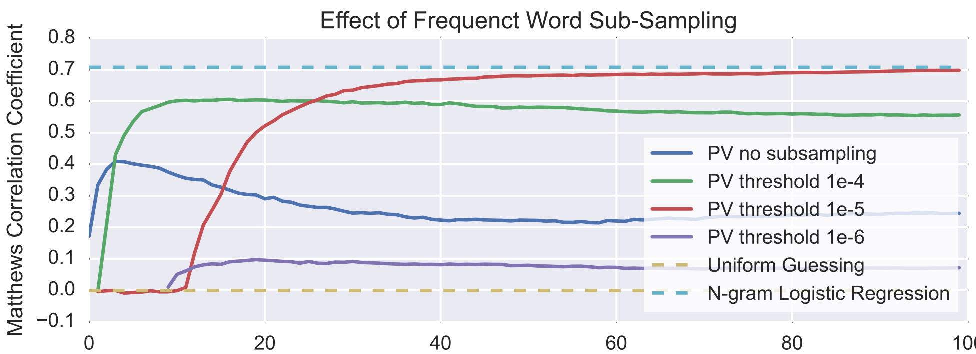


Figure 14: Training of document vectors with different sub-sampling thresholds.

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.

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

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.

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.

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.

PV-DBOW versus PM-DV Both models for paragraph vectors proposed in [Le and Mikolov, 20 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% 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

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.

5.3.5 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.

5.3.6 Paragraph Vectors using context sentences *

5.3.7 Results and Discussion *

5.4 Finding the best Classifier using Vector Space Models

5.5 Advanced and experimental approaches

5.5.1 Inversion of Distributed Language Representations

5.5.2 LSTM Multi-task learner

6 Discussion and Conclusions

6.1 Discussion of Experimental Results

6.2 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)
-

6.3 Contributions

- compare n-gram and doc2vec (?)
-

6.4 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

6.5 Learnings

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

References

[mus, 2016] (2016). mustache . https://mustache.github.io.

[cro, 2016] (2016). Crowdflower. https://www.crowdflower.com.

[mon, 2016] (2016). Mongodb. https://www.mongodb.com.

[nod, 2016] (2016). Node.js. https://nodejs.org/en/.

[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.

[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.