Language modeling for text classification

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: Language modeling for text classification

Date: 16.1.2015 Language: English Number of pages: 6+30

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

Otaniemi, 16.1.2015

Clemens Westrup

Contents

Symbols and abbreviations*

Symbols*

$\mathbf{B}$	magnetic flux density
c	speed of light in vacuum $\approx 3 \times 10^8$ [m/s]
ω_{D}	Debye frequency
ω_{latt}	average phonon frequency of lattice
$\uparrow$	electron spin direction up
$\downarrow$	electron spin direction down

Operators*

$\nabla \times \mathbf{A}$	curl of vectorin $\mathbf{A}$
$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\frac{\partial}{\partial t}$	partial derivative with respect to variable t
$\sum_i$	sum over index i
$\mathbf{A} \cdot \mathbf{B}$	dot product of vectors $\mathbf{A}$ and $\mathbf{B}$

Abbreviations*

kNN	k-Nearest Neighbors
SVM	Support Vector Machine
NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory

Glossary*

one-hot-encoding	sdfsdf
grid search	something

Todo list

1 Introduction*

1.1 Motivation*

1.2 Structure of the thesis*

2 Context *

2.1 Background *

2.2 Corporate Partner *

2.3 Need Statement *

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.

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 im

3 Research Process and Design Development (*)

3.1 Crowdsourced Data Collection (*)

Describe data for-mat

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.

3.1.1 Explorative Paragraph Dataset

Picture of soft-ware 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[1] and it's source code is publicly available on GitHub[2] with it's API documentation hosted online as well[3].

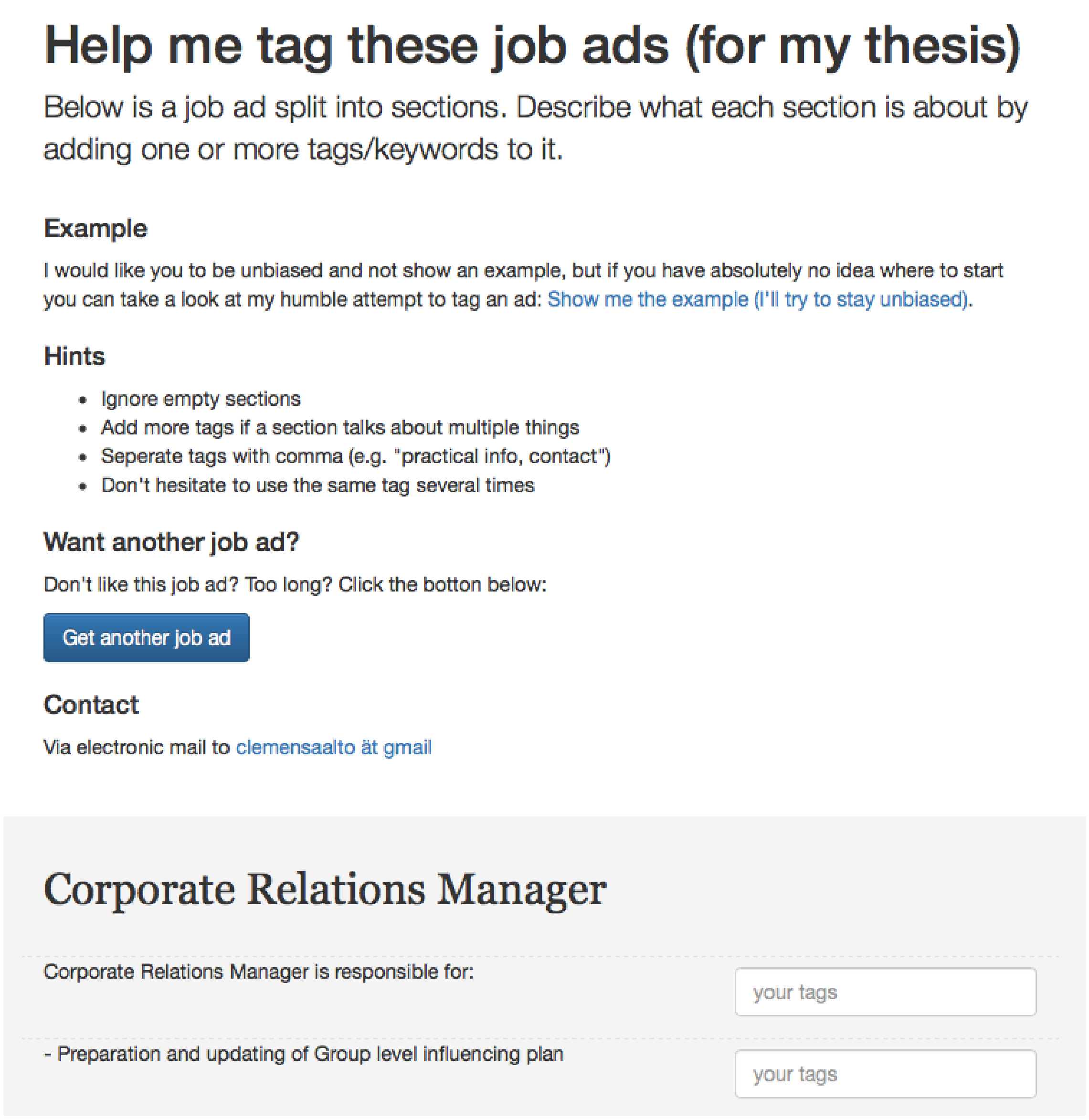
The data generated by using the free-form text description of each job ad and splitting it into paragraphs as can be seen in the software package as well[4].

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.

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


Help me tag these job ads (for my thesis)

Below is a job ad split into sections. Describe what each section is about by adding one or more tags/keywords to it.

Example

I would like you to be unbiased and not show an example, but if you have absolutely no idea where to start you can take a look at my humble attempt to tag an ad: Show me the example (I'll try to stay unbiased).

Hints

- Ignore empty sections
- Add more tags if a section talks about multiple things
- Seperate tags with comma (e.g. "practical info, contact")
- Don't hesitate to use the same tag several times

Want another job ad?

Don't like this job ad? Too long? Click the botton below:

Contact

Via electronic mail to clemensaalto ät gmail

Corporate Relations Manager

Corporate Relations Manager is responsible for:

- Preparation and updating of Group level influencing plan

Figure 1: 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-

3.1.2 Sentence Data Collection

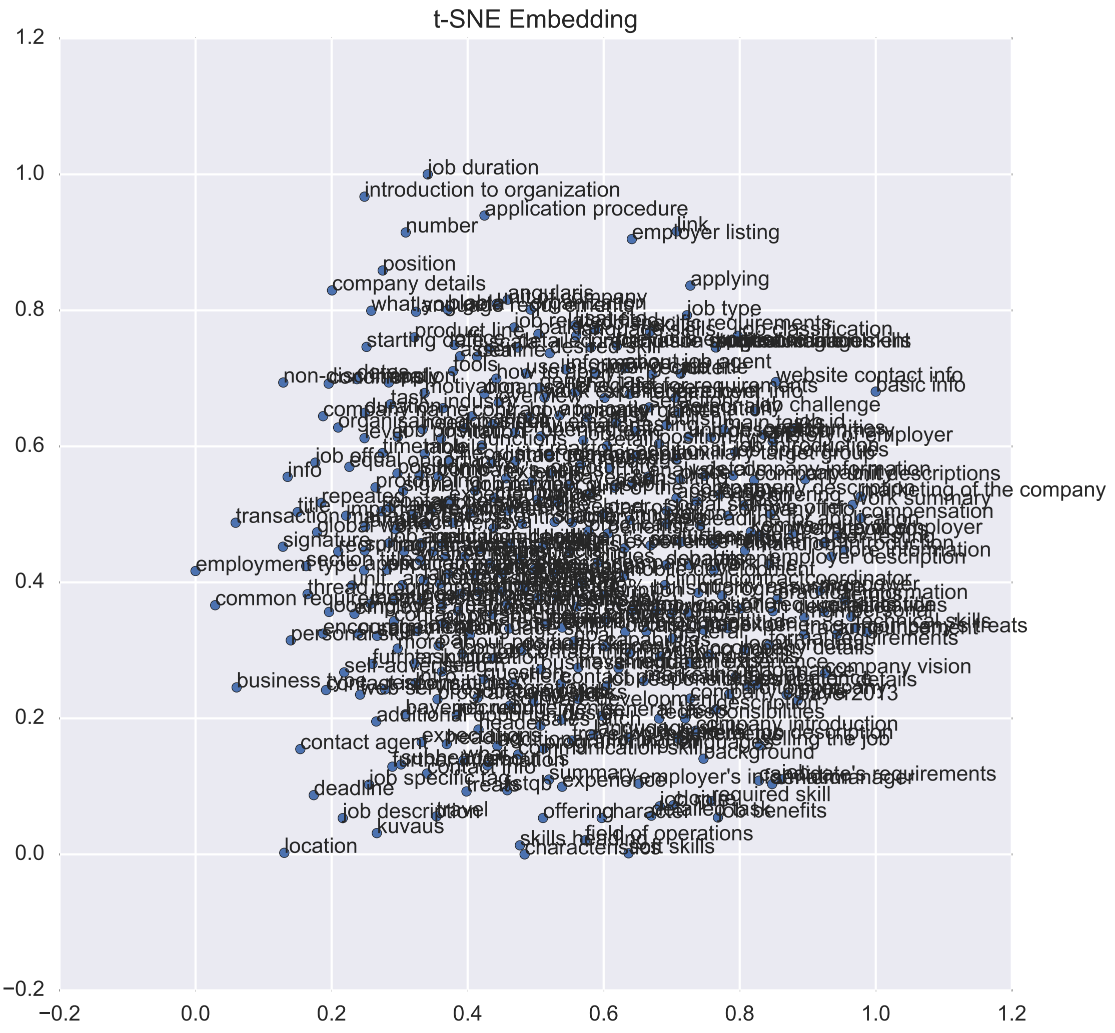


Figure 2: t-SNE Embedding

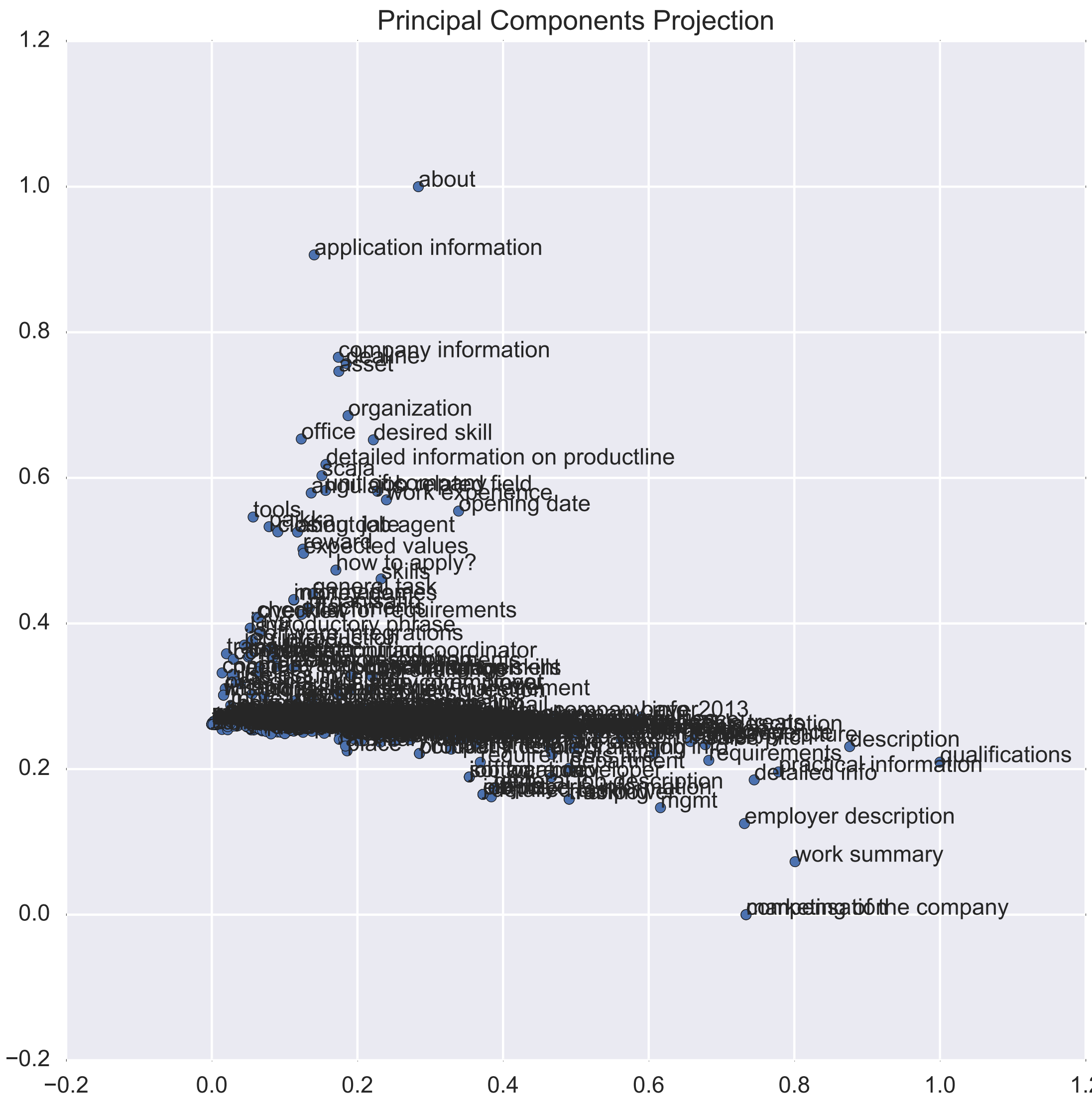
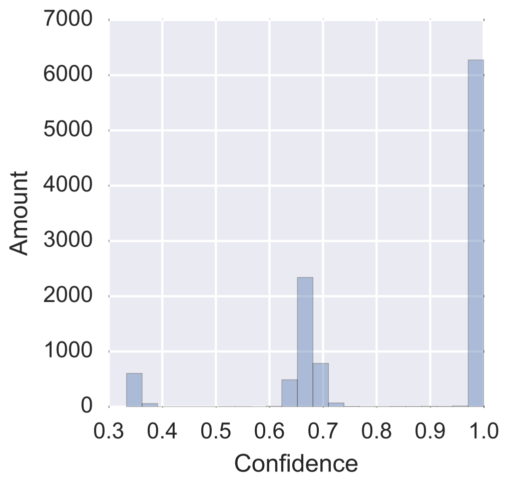
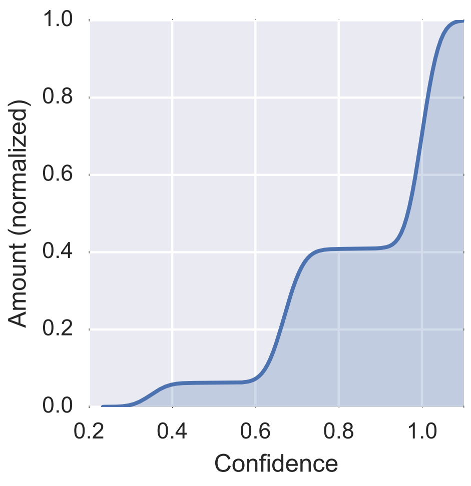


Figure 3: Principal Components Projection



(a) Confidence



(b) Cumulative Confidence

Figure 4: Amount of label judgements versus label confidence of the sentence label data collected via crowdflower

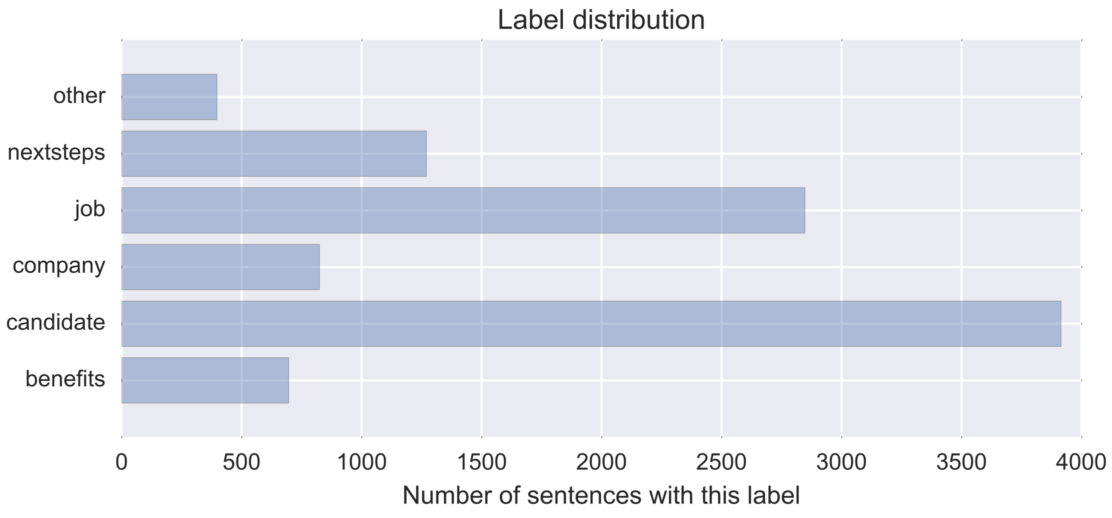


Figure 5: Distribution of labels in sentence data

4 Background

4.1 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.

4.1.1 Binary Classification

In the binary case of classification we are given a single class k and a set of labelled data points $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ 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

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 $(\mathrm{TP} + \mathrm{FP})/(\mathrm{N})$, 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:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

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:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

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).”:

$$\begin{aligned} \text{Informedness} &= \text{Recall} + \text{Inverse Recall} - 1 \\ &= 1 - \text{Miss Rate} - \text{Fallout} \\ &= 1 - \frac{\text{FN}}{\text{RN}} - \frac{\text{FP}}{\text{RP}} \end{aligned} \tag{4}$$

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).”

$$\begin{aligned} \text{Informedness} &= \text{Recall} + \text{Inverse Recall} - 1 \\ &= 1 - \text{Miss Rate} - \text{Fallout} \\ &= 1 - \frac{\text{FN}}{\text{RN}} - \frac{\text{FP}}{\text{RP}} \end{aligned} \tag{5}$$

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:

$$\begin{aligned} r_G &= \pm\sqrt{\text{Informedness} \cdot \text{Markedness}} \\ &= \frac{(\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN})}{(\text{TP} + \text{FN})(\text{FP} + \text{TN})(\text{TP} + \text{FP})(\text{FN} + \text{TN})} \end{aligned} \tag{6}$$

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

$$\mathbb{H}(p, q) = -\sum_{n}^{N} p_n \log q_n \tag{7}$$

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]:

$$\begin{aligned} \mathbb{KL}(p, q) &= \sum_{n}^{N} p_n \log \frac{p_n}{q_n} \\ &= \sum_{n}^{N} p_n \log p_n - \sum_{n}^{N} p_n \log q_n \\ &= -\mathbb{H}(p) + \mathbb{H}(p, q) \end{aligned} \tag{8}$$

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].

Thus, cross-entropy is a measure which is well-motivated from a information-theoretic perspective. It is commonly used in used with generalized linear models and neural networks. In this context we assume that $p \in \{t, 1-t\}$ and $q \in \{y, 1-y\}$ which lets us rewrite the cross-entropy into the following form:

$$\mathbb{H}(p,q) = -\sum_x p_x \log q_x = -y \log \hat{y} - (1-y)\log(1-\hat{y}) \tag{9}$$

(see [Bishop, 2006, Chapter 4.3.2, p. 205] and [Alpaydin, 2014, Chapter 10.7, p. 251]).

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} t_n \ln y_n + (1-t_n)\ln(1-y_n) \tag{10}$$

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.

Effectively this minimizes the This yields a well motivated loss

note on relation to KL-Divergence?

4.1.2 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." 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 this can be defined as follows, with R denoting for Recall and P the Precision.

$$\mathrm{R}_{\mathrm{micro}} = \frac{\sum_i^K \mathrm{TP}_i}{\sum_i^K \mathrm{TP}_i + \mathrm{FN}_i} \qquad \mathrm{R}_{\mathrm{macro}} = \frac{\sum_i^K \mathrm{R}_i}{M} \tag{11}$$

$$\mathrm{P}_{\mathrm{micro}} = \frac{\sum_i^K \mathrm{TP}_i}{\sum_i^K \mathrm{TP}_i + \mathrm{FP}_i} \qquad \mathrm{P}_{\mathrm{macro}} = \frac{\sum_i^K \mathrm{P}_i}{M} \tag{12}$$

And respectively:

$$\mathrm{F}_{1\mathrm{micro}} = 2 \cdot \frac{\mathrm{P}_{\mathrm{micro}} \cdot \mathrm{R}_{\mathrm{micro}}}{\mathrm{P}_{\mathrm{micro}} + \mathrm{R}_{\mathrm{micro}}} \qquad \mathrm{F}_{1\mathrm{macro}} = 2 \cdot \frac{\mathrm{P}_{\mathrm{macro}} \cdot \mathrm{R}_{\mathrm{macro}}}{\mathrm{P}_{\mathrm{macro}} + \mathrm{R}_{\mathrm{macro}}} \tag{13}$$

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:

$$R_k = \frac{\mathrm{COV}(X,Y)}{\sqrt{\mathrm{COV}(X,X)\ \mathrm{COV}(Y,Y)}} \tag{14}$$

Where COV is the covariance function:

$$\begin{aligned} \mathrm{COV}(X,Y) &= \sum_{k=1}^{K} w_k \mathrm{COV}(X_k, Y_k) && (15)\\ &= \frac{1}{K} \sum_{n=1}^{N} \sum_{k=1}^{K} (X_{nk} - \overline{X_k})(Y_{nk} - \overline{Y_k}) && (16) \end{aligned}$$

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 * Another common way of evaluating multi-class classifiers is the categorical cross-entropy which, also known as *multi-class log loss.*
Multi-class Log-loss

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_k) = -\ln p(\mathbf{T} \mid \mathbf{w}_1, \ldots, \mathbf{w}_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk} \tag{17}$$

where $y_{nk} = y_k(\phi_n)$, and $\mathbf{T}$ is an $N \times K$ matrix of target variables with elements t_{nk}.

4.1.3 Multi-label classification *

4.2 Language Modeling

4.2.1 Vector Space Language Models

write vector space models introduction

“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]

Thus each document document vector has the same dimensionality its dimensions can be used as features to be fed into most popular classification metho

explain language modeling as approach for text clas-

4.2.2 N-gram language 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:

$$\text{TF}_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \tag{18}$$

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].

is the term language model right for

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:

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.

citation for first or review paper here?

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 [5].

[5]`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 ??.

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 $\mathrm{IDF}_i = \log_2(N/n_i)$, 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]

4.2.3 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 community but also from open source community (Figure 6 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:

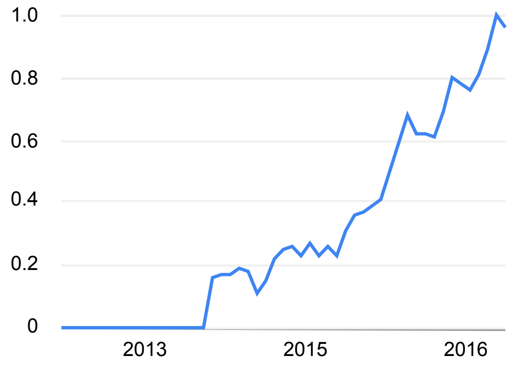


Figure 6: Google Trends statistics on relative search interest in the term "word2vec". Retrieved on 22.05.2016.

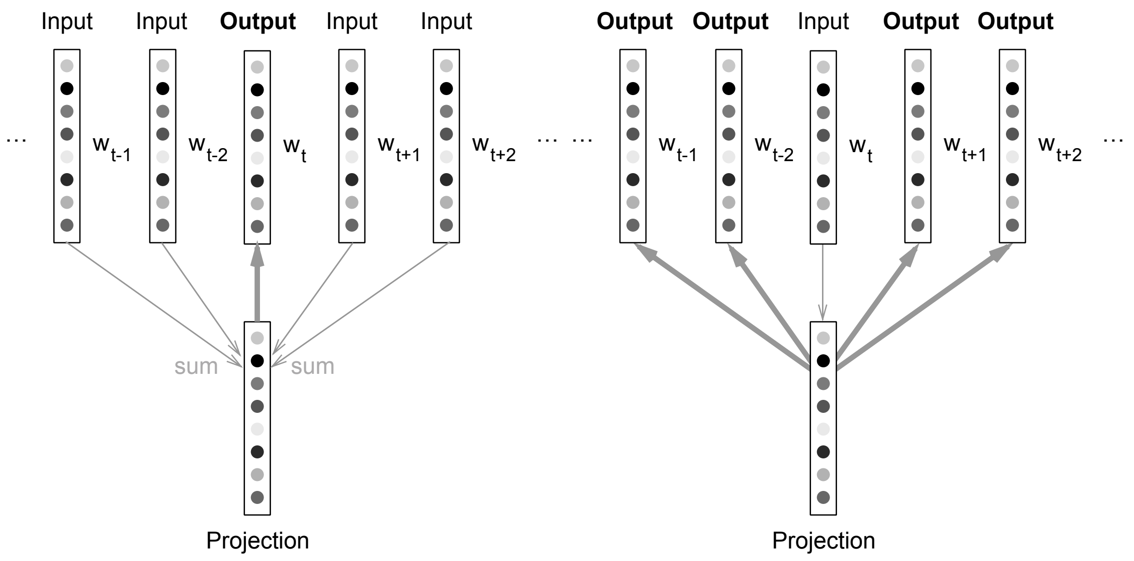
1. Associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$),

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 7 illustrates both models.



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

Figure 7: Architectures for learning continouus distributed word vectors, adapted from [Mikolov et al., 2013a]

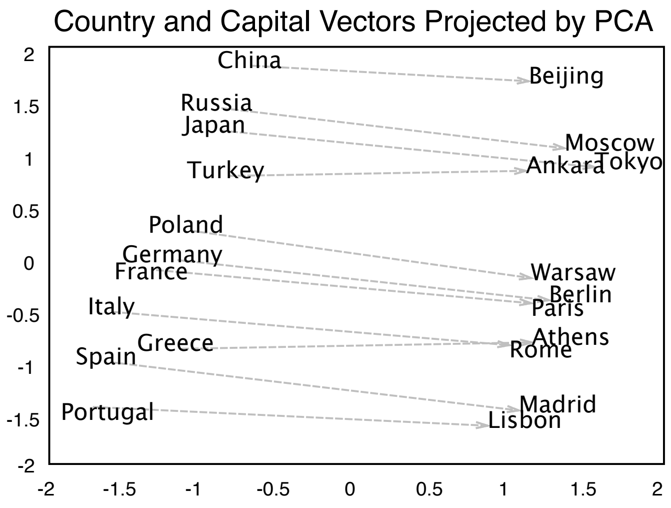


Figure 8: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them.

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 surprising outcome of this research was 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 8 shows a PCA projection of Skip-gram trained vectors of countries and their capital cities.

A notable

mention GloVe

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

Negative Sampling

Hierarchical Sampling

Frequent word sub-sampling

Distributed representations for documents

averaging

parse trees (socher

doc2vec

4.3 Classification Approaches

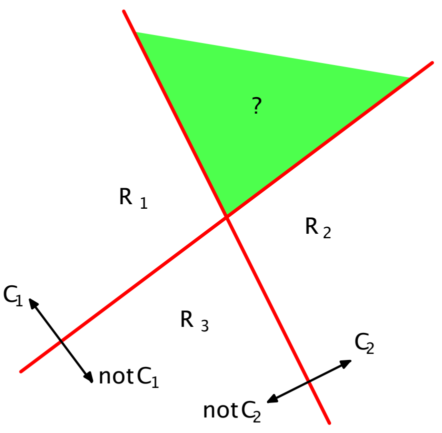
4.3.1 Approaches to Multi-class Classification

A simple approach to multi-class classification is to pose the learning problem as a combination of binary classification problems as described in [Bishop, 2006, Chapter 4.1.2, p. 182]. This can be done by using K separate classifiers, each of which predicts one of the classes against all $K - 1$ other classes, which is known as the *one-versus-the-rest* classification scheme. An alternative approach is to train $K(K-1)/2$ binary classifiers for each possible pair of classes, referred to as *one-versus-one* classification.

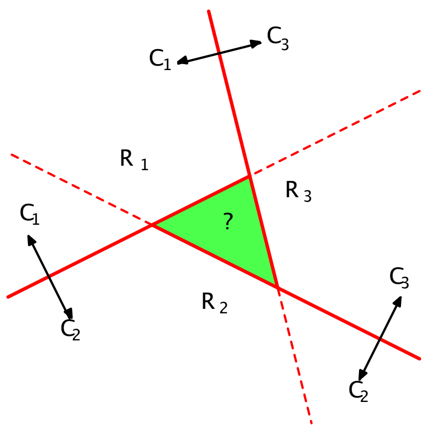
These extensions though have major drawbacks as pointed out by [Duda et al., 1973, Chapter 5.2.2]. As illustrated by 9 both of the classification schemes lead to ambiguous regions in the hypothesis space as their classification is undefined.

[Bishop, 2006]

http://localhost:8888/notebooks/thesis/sandbox/crowdflower-data-collection/extract-data-crowdflower.ipynb



(a) One-Vs-Rest classification scheme



(b) One-Vs-One classification scheme

Figure 9: : Ambiguous regions in the hypothesis space ([Bishop, 2006] Chapter 4, Figure 4.1)

5 Explorative Experiments with Paragraph Dataset

6 Experiments with Sentence Dataset

6.1 Vector Space Models compared

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 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 ?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 multi-class problems and Accuracy.

say why using the sentence dataset here

reference jupyter notebook here

actually discuss time and memory requirements

reference section here

link to logistic regression classifier explanation here

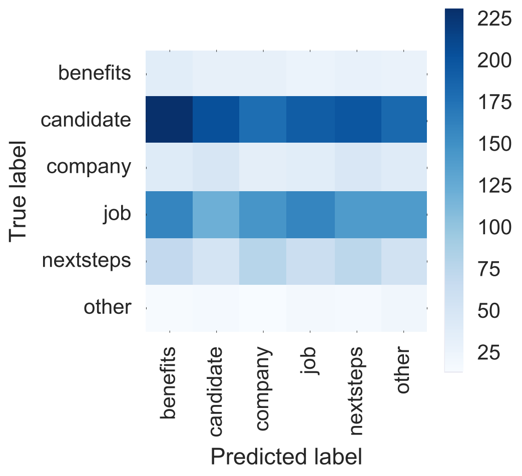
link MCC

6.1.1 Baselines Classifiers: Uniform and Stratified Guessing

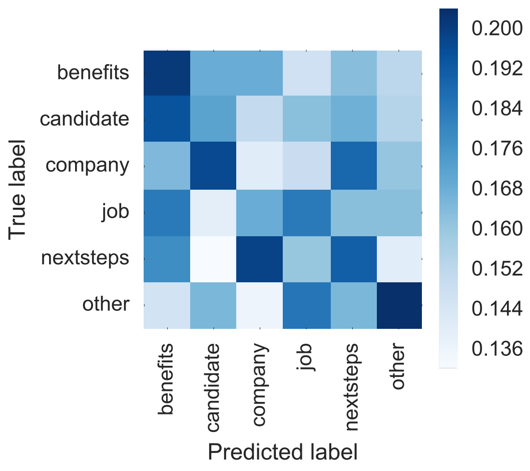
As a baseline for comparing the performance of classification two different guessing strategies were used, namely uniform and stratified guessing. Uniform guessing refers to a predictor that samples from the given classes assuming a uniform distribution whereas stratified guessing takes the label distribution 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 4.1.1). 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 11 shows the confusion matrices for these baseline variants in absolute and normalized form, revealing the properties of these guessing strategies.

6.1.2 N-gram Language Models

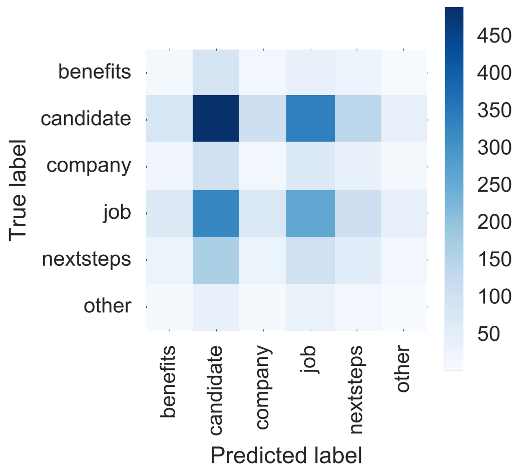
The first class of language models that was investigated for the task of multi-class classification are N-gram models that were explained in Section 4.2.2. As mentioned, in essence this type of model relies on simple statistics which makes for straightforward



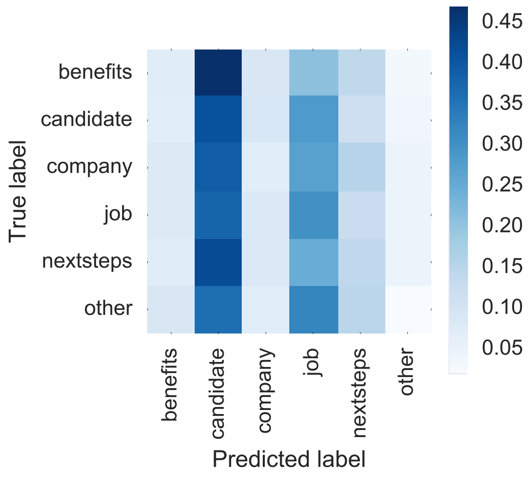
(a) Uniform, absolute



(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 3. The grid search was optimized with regards to its *multi-class log loss* (see section) while using it's resulting model for 5-fold cross-validated classification using Logistic Regression and Naive Bayes.

ref here

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	Log Loss
Word	[1,1]	None	300	No		Yes	-0.667
Word	[1,1]	None	300	No		No	-0.669
Word	[1,2]	None	300	No		Yes	-0.678
Word	[1,2]	None	300	No		No	-0.681
Word	[1,3]	None	300	No		Yes	-0.684
Word	[1,1]	None	300	Yes	L2	Yes	-0.687
Word	[1,3]	None	300	No		No	-0.687
Word	[1,1]	None	300	Yes	L2	No	-0.688
Word	[1,2]	None	300	Yes	L2	Yes	-0.693
Word	[1,2]	None	300	Yes	L2	No	-0.694

Table 4: Top 10 results of grid search over hyper-parameter space as listed in Table 3 using a 5-fold cross-validated Logistic Regression classifier

Find out why the log loss here is so high (or why it's low in the CV)!! is it per class?

6.1.3 Bag-of-Means -word2Vec averaged

6.1.4 Doc2Vec — Distributed representations of documents

Vector Size

Frequeword Sub-Sampling

Hierarchical Sampling

Negative Sampling

Window Size

show influ- ence of each

Type	Range	Stop words	Size	IDF	Norm	Sub-linear TF	Log Loss
Word	1,2	None	300	Yes	l2	Yes	-0.747
Word	1,2	None	300	Yes	l2	No	-0.748
Word	1,3	None	300	Yes	l2	Yes	-0.752
Word	1,3	None	300	Yes	l2	No	-0.753
Word	1,1	None	300	Yes	l2	Yes	-0.755
Word	1,1	None	300	Yes	l2	No	-0.757
Word	1,1	English	300	No	l2	Yes	-0.769
Word	1,1	English	300	No	l2	No	-0.769
Word	1,2	None	300	No	l2	Yes	-0.77
Word	1,2	English	300	No	l2	Yes	-0.772

Table 5: Top 10 results of grid search over hyper-parameter space as listed in Table 3 using a 5-fold cross-validated Naive Bayes classifier

	Training			Validation		
Classifier	Log Loss	Accuracy	MCC	Log Loss	Accuracy	MCC
Logistic Regression	6.602	0.809	0.736	7.324	0.788	0.703
Naive Bayes	8.378	0.757	0.663	8.169	0.763	0.667

Table 6: Performance of each best N-gram model with Logistic Regression and Naive Bayes on the validation data

CBOW versus PM-DV

Evaluating the best hyper-parameter setting

6.1.5 Doc2Vec using pre-initialized weights

6.1.6 Doc2Vec using context sentences

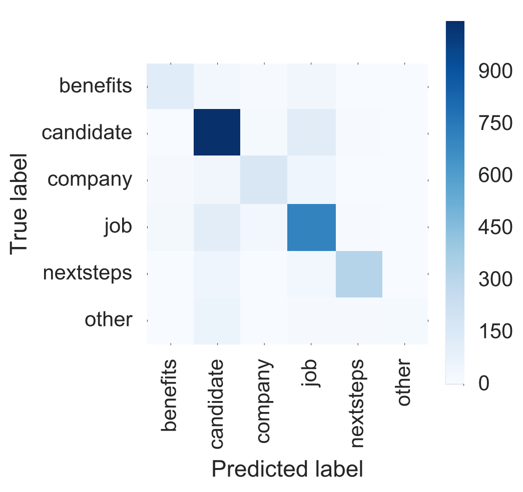
6.1.7 Results and Discussion

6.2 Finding the best Classifier using Vector Space Models

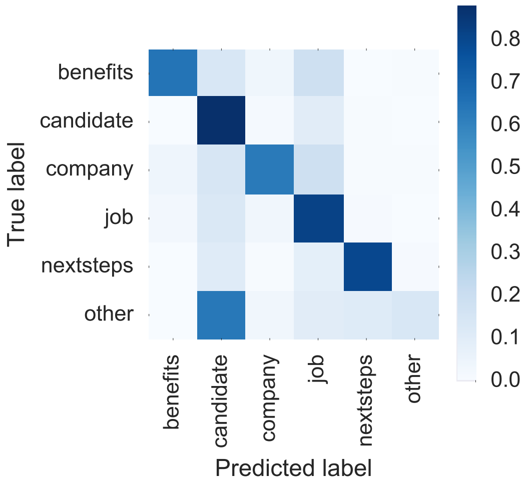
6.3 Advanced and experimental approaches

6.3.1 Inversion of Distributed Language Representations

6.3.2 LSTM Multi-task learner

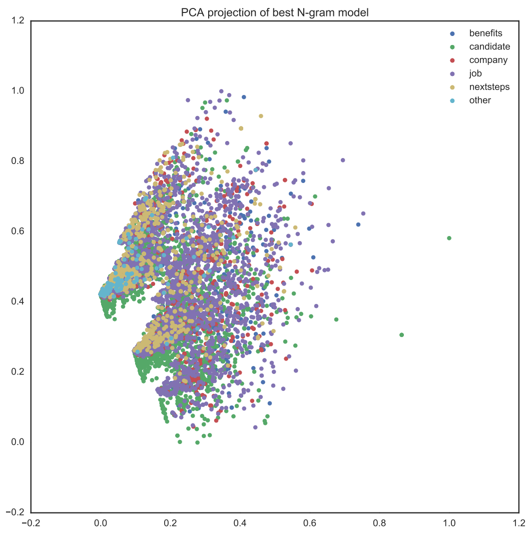


(a) Absolute

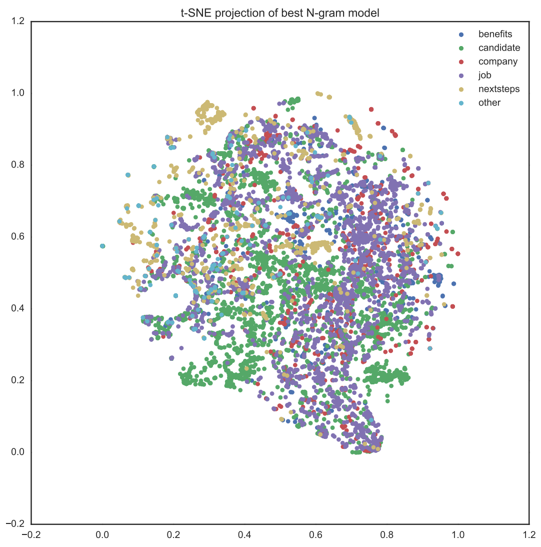


(b) Normalized

Figure 11: Confusion matrix of logistic regression classifier using the best N-gram model found via cross-validated grid search



(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

7 Discussion and Conclusions

7.1 Discussion of Experimental Results

7.2 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)

-

7.3 Contributions

- compare n-gram and doc2vec (?)

-

7.4 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)

7.5 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.

[Duda et al., 1973] Duda, R. O., Hart, P. E., and others (1973). *Pattern classification and scene analysis*, volume 3. Wiley New York.

[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.

[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.

[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., Yih, W.-t., and Zweig, G. (2013b). 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.

[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.

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