

RADBOUD UNIVERSITY NIJMEGEN



Radboud Universiteit

FACULTY OF SOCIAL SCIENCE

Detection of Age Discrimination

SEMI-AUTOMATIC DETECTION OF AGE DISCRIMINATION IN DUTCH JOB ADVERTISEMENTS
THROUGH AUTOMATIC REGEX GENERATION

THESIS MSc ARTIFICIAL INTELLIGENCE

Author:
Anna PILLAR

External Supervisor:
Kyrill POELMANS

Internal Supervisor:
Professor Martha LARSON

February 2020

Abstract

The employment context is the most common domain for age discrimination in the Netherlands, which includes age discrimination in job advertisements. This study proposes a possible architecture for an age discrimination detection approach using regular expressions and machine learning approaches, such as active learning and genetic programming. We conducted two pre studies. First, we identified strengths and weaknesses of regular expression-based age discrimination detection and (2) explored if the latent space spanned by ALBERT word embeddings trained on a job description captures objective age discrimination as defined by the Dutch Equal Employment Act. These findings build the foundation for the proposed age discrimination architecture, which allows the automatic generation of regular expressions with little annotation. Our experiments show that the approach is able to generate valid regexes, but does not yet reach enough generalization to be used as the pure source of regexes for age discrimination detection.

Declaration

I declare this report to be my own original work, and that the contribution of others have been duly acknowledged.

Anna Pillar

Acknowledgments

Throughout the process of writing this thesis, I was lucky to receive support and assistance from several people. I would like to thank:

Textmetrics, for giving me the opportunity to write this thesis in cooperation with them and supporting me along the way. You welcomed me as part of your team, and I am happy to continue my professional journey with you.

Kyrill Poelmans, for being my daily supervisor and always believing in me. I learned a lot during the process of writing this thesis and continue to do so under his supervision.

Martha Larson, my academic supervisor, for giving me the confidence that I have a story to tell and her academic feedback.

Sarah Kemp and Daphne Lenders, who are not only my friends but whose interest in research inspired me along the way.

My mother, my sister, my grandparents and my friends who gave me the mental support I needed and to Ilse, who was there to support me on the last steps, which were the most daunting ones.

Contents

1	Introduction	6
1.1	Ageism	6
1.1.1	Forms of Age Discrimination	6
1.1.2	Ageism as a form of discrimination	6
1.2	Ageism in Recruitment	7
1.2.1	The Dutch Legal Framework	7
1.2.2	Automatic Detection of Age Discrimination	7
1.3	Literate Review	8
1.3.1	Dutch Baseline	8
1.3.2	Text Mining-Based Approach	9
1.3.3	Machine Learning-Based Approach	9
1.4	Limitations to current understanding	10
1.5	Textmetrics	10
1.6	The Current Study	10
2	Data	11
2.1	2014 job description dataset	11
2.2	Discrimination Detection Regexes	12
3	Evaluation of Regex Shortcomings	13
3.1	Introduction	13
3.2	Method	13
3.2.1	Data	13
3.2.2	Annotators	16
3.2.3	Evaluation Measurements	18
3.3	Findings	18
3.3.1	Annotator agreement	18
3.3.2	General Observations	19
3.3.3	Keyword based analysis	20
3.3.4	Context Analysis	22
3.4	Discussion	24
3.4.1	Strength	24
3.4.2	Weaknesses	24
3.4.3	Implications for a Hybrid Approach	24
3.4.4	Limitations	25
3.5	Conclusion	25

4	Language Model	26
4.1	Background	26
4.1.1	Word Embeddings	26
4.1.2	BERT	27
4.1.3	Albert	27
4.2	Methods	27
4.2.1	Data	27
4.2.2	Training BERT	28
4.2.3	Obtaining the word embeddings	28
4.2.4	Visualizing the word embeddings	28
4.3	Findings	29
4.3.1	General View	29
4.3.2	Keyword-Based Analysis	29
4.4	Discussion	38
4.4.1	Clustering based on discrimination labels	38
4.4.2	Context detection	38
4.4.3	Regularities in outliers	40
4.4.4	Implications for a Hybrid Approach	40
4.4.5	Limitations	40
4.5	Conclusion	41
5	Regex Detection Tool	42
5.1	Background	42
5.1.1	Cluster-Based Regular Expression Generation	42
5.1.2	Genetic Programming	43
5.1.3	Using Genetic Programming for Regex Generation	45
5.1.4	Active Learning	47
5.2	Architecture	47
5.2.1	Input	47
5.2.2	Modules	48
5.2.3	Discovery Module	48
5.2.4	Annotation	51
5.2.5	Genetic Search Module	51
5.3	Experiments	54
5.3.1	Data	55
5.3.2	Keywords and key concepts	55
5.3.3	Discovery Module	55
5.3.4	Annotation	55
5.4	Evaluation	55
5.5	Results	55
5.5.1	Intermediate Results Regex Discovery Module	57
5.5.2	Genetic Search	57
5.5.3	Evaluation of the Generated Regexes	57
5.6	Discussion	58
5.6.1	Quality of the generated regexes	58
5.6.2	Possible Improvements	59
5.6.3	Limitations	60
5.7	Conclusion	61
6	Conclusion	62
A	Annotation Guide	63

Chapter 1

Introduction

Even though age discrimination is clearly prohibited by the Dutch Equal Treatment Act regarding age discrimination at the workplace, most age discrimination in the Netherlands happens in the employment context. This starts already in the age pre hiring phase; over 40,000 Dutch job descriptions contain age discriminating phrases [1].

This chapter gives an overview on the topic of age discrimination, with focus on the Dutch employment context. Afterwards, we present current approaches to detect age discrimination and conclude with our research questions.

1.1 Ageism

Age discrimination in the widest sense refers to the bias and prejudice of people based on their age [2]. By now, however, it is more commonly used to describe the discrimination of the old [3]. In the Netherlands, discrimination based on age has increased recently, mainly in the employment field [4]. For people over the age of 45, age discrimination is the most reported form of discrimination in the work field, with people aged 55-64 years being the most discriminated group [4]. The majority of people in this age group do not expect to find a job any more [4], resulting in unemployment and financial challenges. But there are more negative effects than just not finding a job, such as health issues, decrease in well-being and social isolation [5, 4].

1.1.1 Forms of Age Discrimination

Age discrimination occurs in two main forms, *objective* and *subjective* age discrimination. *Objective Ageism* is the ageism which is defined through legal frameworks to protect the vulnerable group from discrimination [6]. Objective ageism is illegal, in contrast to *subjective*, or *perceived ageism*. This form of ageism describes all instances of discrimination which do not fall under the legal definition of ageism, but may nonetheless discriminate against individuals [6]. An example would be prejudice in an everyday life situation, e.g. assuming that an elderly person needs help with a physical task even though there is no clear evidence that they do.

In addition to these two distinctions, ageist actions can be separated into two categories. Acting ageist, no matter of objective or subjective can happen purposefully, which is known as *explicit ageism*. If, however, the actors is not aware that their action is ageist, one speaks about *implicit ageism* [7].

1.1.2 Ageism as a form of discrimination

Even though ageism is considered to be one of the big “-isms”, next to racism and sexism, it holds several special properties which differentiates it from other forms of discriminations [2]. For this type of discrimination, implicit discrimination plays a significant role, as most people are unaware about the

extent of age discrimination or even the concept itself. Most age discrimination is, in fact, implicit, since it is a culturally accepted form of discrimination [7]. Further, it holds the unique property that it will affect everyone at one point in their lives [8]. It is the only form of discrimination in which in-group members (young people) transition to out-group members (old people) [9]. Exactly this transitioning from in-group to out-group throughout time may explain the self-targetness and internalization of age discrimination. Often, old people diminish themselves with ageist beliefs and “becoming old” is generally considered to be something bad, something one should try to avoid [7]. These properties are some reasons why it is so hard to detect age discrimination. It is present in every area of life, however, specifically for the Netherlands, the employment context is both the common area of life for age discrimination [4].

1.2 Ageism in Recruitment

Discrimination in the workplace already begins in the pre-hiring phase, i.e., in job advertising. The wording of any given job advertisement can be off-putting to a potential candidate. Applicants are more likely to apply to companies that match their social identity [10]. For example, older adults are less likely to apply for positions wherein the advertisement uses language such as “digital native”, or other terms that imply a younger candidate is desired. Further, some advertisements use explicit terms such as “student position”, “early graduate position”, etc. Older adults are less likely to apply to these positions than younger adults. Further, a recent study found that the occurrence of skills which are connected with stereotypes against the elderly (people older than 55) negatively correlate with the age of the candidates which are invited to an interview. If, e.g. physical fitness is required for a job, older candidates are less likely to be even invited to an interview even though the hiring company has no way of knowing about the physical abilities of the applicant. In 2019, around 100,000 job descriptions in the Netherlands contained ageist statements, e.g., advertising jobs for specific age groups or specific targeting of jobs to students [1]. Therefore, job descriptions which include ageist statements either discourage older people (>55 years) to apply or, if they do, they are less likely to be invited to an interview or hired.

1.2.1 The Dutch Legal Framework

The Dutch Equal Treatment Act regarding age discrimination at the workplace (*Wet gelijke behandeling leeftijd bij de arbeid*) differentiates between two forms of objective age discrimination. For the domain of job descriptions, *Direct* age discrimination entails direct age requirements which are not justified. This includes numerical age mentions, age groups, generation mentions and the terms “young” and “old”. Examples of justifications can, e.g., be that applicants must be older than 18 due to security reason. *Indirect* age discrimination means the mentioning of inherently neutral terms, which, however, imply a certain age (group), such as the term “student”. Exceptions are terms such as “junior” and “senior” as they commonly refer to the skill level and not the age level of an applicant [1].

1.2.2 Automatic Detection of Age Discrimination

Given the vast amount of job advertisements which are nowadays available online, automatic approaches, i.e., automatic age discrimination detectors, are needed to detect age discrimination. These detectors can be classified either as *warning systems* or *research tools* [11]. Warning systems are systems that are targeted towards authors of job descriptions and should be optimized for precision. Research tools on the other hand should be developed with an eye to recall, to assure that these tools can be used to survey the extent of the discrimination in a large corpus. However, the focus on one measure for a specific approach should not mean that the other score is neglected and a high F-score should generally be striven for.

1.3 Literate Review

The most common approaches to automatically detect age discrimination is to this point done with keyword matching [12], but there is also a first approach which utilizes machine learning techniques to age discrimination detection [12]. In the following, we will present three different approaches to age discrimination detection in job description, which motivated choices for the current study.

1.3.1 Dutch Baseline

The current Dutch baseline on detecting age discrimination [11] is an example of a keyword matching approach. Fokkens et al. constructed a list of regular expressions which can be used to detect age discriminating statements in text to survey the amount of age discrimination in Dutch job advertisements. Their list of regexes was handcrafted after reading several job descriptions and identifying common patterns of age discriminating phrases. Following the Dutch legal framework, they split this list of regexes in two groups, direct and indirect discrimination. For each of these discrimination types, they constructed two different types of regex list, *strict* regexes and *flexible* regexes.

Strict regexes are formulated in a way that they require a word-by-word match to any of the discrimination regexes. An example for this would be the regex,

$$(?P<relstr>je\s+bent\s+een\s+ student)$$

which matches all sentences containing the phrase “je bent een student”. It would, however, not match phrases which are more elaborative, as e.g. “je bent een HBO-student”. Phrases like this can be captured with the *flexible* regexes, which add the regex wildcard character “.” to the regular expression. The flexible counterpart to the above given sample would be:

$$(?P<relstr>je\s+bent\s+een\s+.{0,30}student)$$

which allows 0-30 characters between the word *een* and *student*. However, this comes with the danger that sentences are matched which are not age discriminating. One example for this would be “je bent een docent voor HBO-studenten”, which matches the flexible regex but is not a discriminating statement.

The study was conducted for a set of job descriptions from two different years, once for job descriptions from 2017 [11] and once for job descriptions from 2019 [1]. They reported recall and precision of their regular expression for each year, together with an analysis explaining the shortcomings of them. This evaluation was done by applying the regexes to a manually labeled dataset composed out of job descriptions from the respective year.

We summarize their evaluation in Table 1.1.

	Strict algorithm		Flexible Algorithm	
	Precision	Recall	Precision	Recall
Direct Discrimination	90.7%	30.1%	87.0%	36.7%
Indirect Discrimination	92.0%	60.0%	94.3%	89.3%
Combined (direct and indirect)	92.8%	48.1%	94.5%	75.7%

Table 1.1: Precision and Recall for the automatic age discrimination detection approach for the Dutch baseline [1]

The results show that while a somewhat high precision was reached, the recall is rather low, especially for direct discrimination. Regarding the reported recall, they state that the sample set itself is not fully representative and hard age discrimination samples were excluded. Therefore, the actual recall is most likely lower than reported.

When comparing the results of both years with each other, they noticed that in the 2019 dataset, the number of false positives (phrases that were marked as are discriminating but actually are not) is

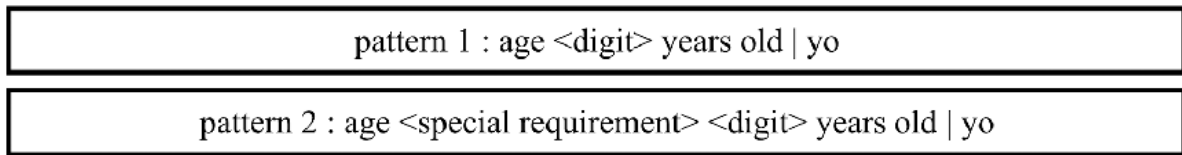


Figure 1.1: Word Pattern used for age discrimination detection in Indonesian job ads [13]

more frequent for the keyword “*opleiding*”. It appears that in 2019, more job descriptions advertised a training on a job with a phrasing that was captured with the regular expression that was meant to match phrases which require the applicant to be still in education. Further, the evaluation also revealed that some keywords are missing and complete types of discriminations remain undiscovered. One example for this is the keyword “*young*”, often used in context of “*young professionals*”. One hypothesis is that these undiscovered keywords may be the biggest reason for the low recall of the regexes. They conclude that more research into the wording of age discrimination is needed to accommodate for this.

1.3.2 Text Mining-Based Approach

Another example of detecting (age) discrimination utilizes text mining techniques to survey several forms of discrimination in Indonesian job descriptions [13]. The analysis was done on a set of Indonesian job advertisements, scrapped from the web. To detect discrimination in these ads, the authors create a dictionary of discriminating keywords and extract all job descriptions which contain any of these keywords from the corpus. Since these keywords are not sufficient to classify a job description as age discriminating (e.g., the keyword *age* may also be used in a non-discriminating context), they construct so-called *Word Pattern Templates*. For this, they extract n-grams of the words behind the keywords and group them into different categories and build Word Pattern Templates with the discriminating samples. Figure 1.1 shows two examples of such a word pattern template. Eventually, these word pattern templates are then used to label the corpus for the discrimination analysis. In contrast to the Dutch Baseline, no evaluation of the approach in terms of precision and recall of the detection is given.

1.3.3 Machine Learning-Based Approach

Lastly, a recent study successfully utilized machine learning techniques to detect ageist stereotypes in job descriptions [12]. As with the previously presented approaches, the aim of their study did not purely lay in the detection of age discrimination. Instead, they research the question if job descriptions can predict the hiring strategy of companies in terms of the age of the applicants. For this, a small resume study was conducted, for which fictional but realistic resumes of candidates for different age groups were constructed. The researchers used these resumes to respond to real job openings and evaluated if there is any correlation between the semantic similarity of ageist stereotypes in the job description and the ages of the candidates which are invited to an interview. To calculate this semantic similarity, they utilized a machine learning technique, known as *Word2Vec*, which we will discuss in more detail in Chapter 4. In short, this approach allows the researcher to give each trigram of words in a job description a score which represent the semantic similarity to a stereotype. For each of the job descriptions, they compute this similarity score for each trigram to each stereotype and create a probability distribution over each stereotype for the job description. The 95th percentile of this distribution serves as the final stereotype similarity score for the job advertisement and is used to research the correlation with the age of the applicants invited to an interview. As with the text mining approach, this research also does not report any performance scores of their approach. Given the nature of this score, however, an evaluation as done by Fokkens et al. would have been more difficult, as they provide an artificial score which may be challenging to verify.

1.4 Limitations to current understanding

Discrimination, and age discrimination in particular, is complex. Detecting ageist statements in text is not a trivial task [7] and even if there are clear legal guidelines, language is complex, vague and discriminating statements are not always obvious. Keyword-based approaches provide a first good baseline to the problem of age discrimination detection, but there are arguments that they are not enough to capture the complexity of discrimination in texts [12]. Recent advances in the field of Machine Learning have also benefitted Natural Language Processing, and more and more approaches can infer about the semantics of a sentence rather than just processing its syntactically structure. The study of Burn et al. [12] shows that these techniques can also be successfully applied as research tools to quantify the similarity of job descriptions to ageist stereotypes.

However, these are new developments, and it yet needs to be investigated to which extent these tools can be used in the domain of age discrimination detection.

1.5 Textmetrics

This Thesis was written in cooperation with Textmetrics and supported this research with data and annotation capabilities. Textmetrics is a Dutch company, located in Arnhem. They focus on aiding recruiters in the process by giving them feedback on the inclusiveness of their job advertisements. This includes feedback on gender bias, readability, accessibility and also age discrimination. Textmetrics will use the findings of this thesis to improve the objective age discrimination feedback they are giving their customers.

1.6 The Current Study

This thesis investigated the current Dutch method of detecting age discrimination as well as possible machine learning approaches to propose a hybrid model for detecting age discrimination in a context with limited annotation options and the need for high explainability of the detection results. The focus will be put entirely on objective age discrimination as defined by the Dutch Equal Treatment Act, which includes regulations for direct and indirect age discrimination.

We approached this question through three sub studies.

1. Through a qualitative evaluation, we identified a list of shortcomings but also benefits beyond the reported recall and precision of the current regex-based Dutch baseline for age discrimination detection.
2. A second qualitative evaluation was done to observe if latent space spanned by word embeddings obtained by training a BERT model on a job description corpus captures objective age discrimination, for both direct and indirect discrimination.
3. Lastly, the results of both studies were used to generate a list of requirements for a hybrid age discrimination approach, which enables detection of age discrimination when only limited annotated data is available while maintaining explainable results.

Considering the limited research on utilizing machine learning techniques to detect age discrimination, this thesis is approached with a discovery-driven approach. The architecture we propose to answer the last sub question should be considered a first prototype. Its architecture was informed by the findings of the first two sub questions and provides a contribution to a small research corpus.

Chapter 2

Data

2.1 2014 job description dataset

To the best of our knowledge, there is no commonly available annotated dataset for any form of discrimination detection in Dutch job description. We, therefore, had to obtain an unlabeled dataset for this study. The dataset we used is a collection of 1,605,675 (1,225,062 after duplicate removal) Dutch job advertisements which were crawled by a Dutch company, Jobdigger in 2014. Jobdigger allowed the usage of this proprietary dataset for research purposes within Textmetrics as well as the publication of findings obtained based on this dataset. We refer to it as the *2014 job description dataset*.

The whole dataset contained job descriptions with 215514 different occupation types. Table 2.1 gives an overview of the ten most frequent occupation types. We excluded all samples which called for volunteers or interns, as these do not fit the context of this study. Further, we excluded all non-Dutch job advertisements, using the `langdetect` package for python. Since the dataset was obtained through web crawling, some cleaning had to be applied to remove all HTML elements. Afterwards, the individual job descriptions were split into sentences. This was done using the spaCy sentence tokenizer for Python¹.

The final dataset contained 5,865,749 sentences (11,537,167 before removing duplicate sentences).²

¹<https://spacy.io/api/tokenizer>

²The high number of duplicate sentences can be explained by the fact that job descriptions from the same company often contain an overlap. Further, one and the same job could be advertised with only the location differing. These job descriptions would not be detected by the full-text duplication scan but only on a sentence to sentence basis

Occupation type	Number of job descriptions
Vrijwilligers	11066
Accountmanager	7972
Stage	6702
Medewerker	5468
Stagiair	5346
Vrijwilliger	5287
Productiemedewerker	4874
Magazijnmedewerker	4612
Projectmanager	4296
Verpleegkundige	4213

Table 2.1: 10 most frequent occupations in the 2014 Job Description Dataset

Direct Discrimination	Indirect Discrimination
jong	bijverdien
min	bijbaan
min en max/hooguit	starte
leeftijd vanaf	ervaring
leeftijd tot	stap
leeftijd van tot	start
leeftijd bepaald	lesrooster
generatie	baan
oud	stud
ouder niet dan	opleiding
max	jong groep passen
babyboom	jong deel uit maken
tussen	jong groep versterken
	schoolverlater

Table 2.2: Discrimination keywords for the direct and indirect discrimination regexes

2.2 Discrimination Detection Regexes

In addition to the job description dataset, we also made use of a list of regexes. We introduced these regexes already in the last chapter as the Dutch Baseline for age discrimination detection, developed by Fokkens et al. [11]. This baseline is publicly available on GitHub³.

As described previously, the regexes are split into four lists, a list of flexible and strict regexes for each discrimination type. Additionally, each list of regexes is structured by the discriminating keyword they are addressing.

It was necessary to change some of the regexes constructed by Fokkens et al., as not all of them were valid regexes and contained minor mistakes. Further, an additional regexes were matched to target the word “schoolverlater”. The list of changes is proprietary to Textmetrics. Table 2.2 lists the discriminating keywords for both direct and indirect discrimination.

³Age Discrimination Baseline by [11] <https://github.com/cltl/AgeDiscriminationBaseline/tree/master/patterns>

Chapter 3

Evaluation of Regex Shortcomings

3.1 Introduction

The current regex-based Dutch baseline for age discrimination detections suffers from a rather low recall [1]. Fokkens et al. hypothesize that this is caused by missing discriminating keywords, i.e., the list of regexes does not cover all possible areas of discrimination. To improve the current baseline in terms of recall, a qualitative analysis was conducted with the goal to detect strength and weaknesses of the current regex approach. We expected that these results together with the reported quantitative results from the original study [1] will inform the development of an improved age discrimination detector. We particularly focused on the False Negative samples (i.e., discriminating samples which are not detected as such), as the recall can be improved through reducing the number of False Negative samples.

For this evaluation, we differentiate between four different kinds of regex matches. A *true positive match* (TP) means that the regular expression matches a string which contains age discrimination, sentences that match a regular expression but do not actually contain age discrimination we refer to them as *false positive* (FP) matches. *False negative* (FN) matches are those sentences that contain age discrimination but are not detected by the regexes, and *true negative* (TN) matches are those which do not contain age discrimination and do not match any of the regexes.

3.2 Method

To evaluate the performance of the current baseline beyond recall and precision, we first create a small annotated corpus of job descriptions and try to identify common properties of False Negative samples. Further, we also explore how much the direct context (i.e., the words surrounding a keyword) reveal about possible discriminating usage of the keyword.

3.2.1 Data

We use the previously introduced *2014 job description dataset*, which consists out of a collection of sentences taken from online job advertisements, and sample it to satisfy the specific needs for this study (discussed below). Since the dataset is unlabeled and contains too many sentences to be annotated feasibly, a subset had to be selected and annotated by employees of Textmetrics, the company supporting this research.

The regexes used for this evaluation are based on the regexes from the study Fokkens et al. conducted, they are publicly available at <https://github.com/clt1/AgeDiscriminationBaseline>. As already mentioned in Chapter 2, the regexes are split in two sets, one targeting direct discrimination and the other targeting indirect discrimination. Each of these sets is organized by keywords, i.e. in each file, a number of regexes target the same keyword.

```

===jong===
(?R<prestr>.*)(?R<relstr>w(ij|e)[\s]*zoeken[\s]*(een\s)?.{0,30}jong(e|eren|ere)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>zoeken\s+w(ij|e)\s+(een\s)?.{0,30}jong(e|eren|ere)?\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>(ij|e)\s+zijn\s+op\s+zoek\s+naar\s+(een\s)?.{0,30}jong(e|eren|ere)?(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>zijn\s+w(ij|e)\s+op\s+zoek\s+naar\s+(een\s)?.{0,30}jong(e|eren|ere)?(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>vacature\s+voor\s+(een\s)?.{0,30}jong(e|eren|ere)?(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>ben(t)?\s+(ij|j|e|u)\s+(een\s)?.{0,30}jong(e|eren|ere)?(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>(ij|j|e|u)\s+bent\s+(een\s)?.{0,30}jong(e|eren|ere)?(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>ben(t)?\s+(j|i|j|e)|u)\s+.{0,30}\d\d\s+jaar\s+of\s+jong(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>(ij|e)|u)\s+bent\s+.{0,30}\d\d\s+jaar\s+of\s+jong(!\s(\s)?van\geest)\s)(?R<poststr>.*)
(?R<prestr>.*)(?R<relstr>onze\s+voorkeur\s+.{0,30}(gaat\s)?uit\s+naar\s+(een\s)?.{0,30}jong(e|eren|ere)?(!\s(\s)?van\geest)\s)(?R<poststr>.*)

```

Listing 3.1: Exerpt from the regexes used by Fokkens et al. [11]

Listing 3.1 shows an example of a collection of regexes which are all targeted around the keyword “jong”. Chapter 2 gives a detailed overview over the keywords which can be found in the list of regexes. Note that some keywords are actually composed out of several words, such as “leeftijd vanaf”, but as they are used as sub-categories to the keyword “leeftijd”, we refer to those two or more worded keywords still as a keyword.

Selection of samples

The choice of selected samples for annotation was motivated by three main objectives. The first and overarching objective for the annotation process were the resources available for annotation. Our annotation set size was limited to 3000 samples in total, which has implications on the other two objectives.

Secondly, we aimed to include as many False Negative samples as possible in our annotation set. This was to ensure there would be enough samples for a qualitative analysis of discriminating sentences that the regexes would fail to detect. This meant that we purposely sampled the data in a non-representative way to increase the occurrence of False Negative data for our analysis. Hypothesizing that the occurrence of a keyword increases the likelihood of the sentence being age discriminating, we took a subset of the data which contains one of the age discriminating keywords but does not match any regex. By excluding all sentences which match a regex, it was assured that all sentences in this set which are truly age discriminating are False Positives (undiscovered by the regexes). The subset of sentences containing keywords was rather big, and the general the frequency of age discrimination is only at about 3.14% for indirect discrimination in 2020[1]. We, therefore, decided to allocate the biggest part of our sample count to satisfy this motivation, as the expected False Negative count would be significantly smaller than the amount of True Negative samples this subset entails. This meant that 2000 samples of the described subset were included into the annotation set. The exact composition of this will be described in the next section.

The third and last objective was to investigate those samples which contain a regex match. On the one side, this would allow investigating if there are any common properties of False Positive samples. Further, the group of True Positive samples could provide further information on the nature of age discrimination which can be detected with regexes. The total amount of regex matching samples included in the annotation set was set to be 1000.

With this distribution, we aim to get enough data for each of the relevant categories (False Negative, True Positive and False Positive) to analyze the strength and weaknesses of the regex-based Dutch Discrimination Detection Baseline.

The next section will give a more detailed account of the composition of the annotation dataset.

Composition of the Annotation Data Set

For each of the 30 keywords, we created subsets of the dataset composed only out of sentences which contain the respective keyword. These subsets were then annotated with the regexes, indicating if a sentence matches one or more regexes or not.

Firstly, for each of the keywords, samples were selected which do not match any regex (Objective 1). For this, we split the keywords into two groups. The first group contained keywords which are split into smaller subgroups, e.g. there is a *leeftijd*, *leeftijd vanaf*, *leeftijd tot* and *leeftijd van tot* subset. The second group contained all the other keywords and phrases. From the first group, 50 non-regex matching

keyword	Samples total	Matching any regex
jong	103	41
geboortejaar	8	0
generatie	46	0
babyboom	3	0
tussen	101	37
tot	170	19
bijverdienen	39	2
bijbaan	61	6
ervaring	135	54
stap	85	59
start	209	121
lesrooster	37	0
baan	145	66
stud	268	175
opleiding	219	127
schoolverlater	147	82
leeftijd	145	4
leeftijd_vanaf	46	13
leeftijd_tot	52	2
leeftijd_van_tot	46	1
leeftijd_bepaald	51	0
min	240	90
min_en_max/hooguit	11	0
oud	201	71
ouder_niet_dan	9	3
jong_groep_passen	50	8
jong_deel_uit_maken	107	76
jong_groep_versterken	1	0
No Keyword	248	0

Table 3.1: Keyword/phrase occurrence frequencies for the to be annotated dataset

samples were selected. For the second group of keywords, up to 150 non-matching samples were selected. Not all keyword subset had enough samples which satisfied this condition. For those, the maximum amount was selected.

Table 3.1 gives an overview of the total distribution of keywords in the annotation data set. Note that some keywords tend to occur together in one sentence. The annotation set was controlled for duplicates, but a sentence selected for one keyword may also contain another keyword, which was not counted towards the count of the accidentally included keyword. We retrieved a total of 1703 keyword containing sentences. The remaining 297 sentences for the 2000 unmatched sentences were filled with randomly selected sentences which do not contain any keyword. These did not match any regex, as each regex contains a keyword.

Secondly, for each keyword, up to 75 samples which match a regex were selected (Objective 2). As with the above selection, several keyword subsets had less than 75 samples matching a regex. Given the varying availability of sentences matching regexes, selecting samples up to this number provided us with a sample set size of 983 sentences and thus nearly reaching our target of 1000 regex matching sentences.

This results in a sample set size of 2983 samples, with 2000 samples not matching any regexes and 983 samples matching a regex.

3.2.2 Annotators

In total, 7 annotators annotated the dataset. The annotators were split in two different groups. Each group annotated the whole annotation set once. This means that no annotator has seen all samples, but each sentence was annotated by two different people (from two different groups). This decision had to be made due to limited resources. Five of the annotators formed group one, two group two. All the annotators are not part of the sensible age group (all significantly younger than 65 years) and there was no age discriminating expert among the annotators. However, a few employees have had more contact with the topic of age discriminating due to Textmetrics recent focus on age discrimination detection. For all others, the first real introduction into the specifics of age discrimination came from the annotation guide.

Annotation Tool

The annotation of the dataset was done in Doccano [14], an annotation software, which offers an easy-to-host web environment for document annotation.

Annotation Labels

Each annotator received an annotation guide and log-in credentials to the platform. The annotators had to decide between five different labels.

- Direct Discrimination
- Indirect Discrimination
- Other
- No Discrimination
- Unsure

The labels *Direct* and *Indirect Discrimination* follow the definition from before (see Section 1.2.1). Samples labeled with *Other* are discriminating samples, but seem not to fit into the direct or indirect category. This category was introduced to allow the annotators to highlight edge cases which were not accounted for during the creation of the annotation guide. This could be previously not discovered keywords, clearly age discriminating phrases based on the tone of voice, but also sentences which give an objective justification for the age discrimination. The full annotation guide can be found in Appendix A.

Annotation Processing

After the annotation, the agreement between the annotators was checked and all samples for which the annotators disagreed on the label were excluded.

Samples were considered to have agreement between annotators if they annotated the sentence with the same label. This was the case for 2195 sentences. For 779 sentences, annotators disagreed on the label, these were excluded from the dataset. The decision to exclude these samples rather than to resolve the conflict was made because there was no expert on age discrimination available who would be qualified to determine the true label of the sample. Further, one sentences was excluded which only received one annotation. The remaining 11 samples were excluded because one of the annotators labelled them as “unsure”. Here, we again decided for discarding the sentences, as there was no expert present and we wanted to make sure only samples which received a clear labeling from both annotators would be included.

Table 3.2 gives an overview of the label distribution of the dataset, Table 3.3 an overview of the keyword occurrences in the annotated dataset. Note that, once again, one sentence may contain more than one keyword. This set of sentences is what will be from now on referred to as the *annotated data*.

Label	Samples
No Discrimination	1262
Indirect Discrimination	642
Direct Discrimination	254
Other	19
Internship	18

Table 3.2: Overview of the labels in the annotated dataset

Keyword	Samples
<no keyword>	235
baan	182
babyboom	2
bijbaan	141
bijverdienen	34
ervaring	266
geboortejaar	2
generatie	48
jong	134
jong_deel_uit_maken	17
jong_groep_passen	19
jong_groep_versterken	1
leeftijd	90
leeftijd_bepaald	28
leeftijd_tot	35
leeftijd_van_tot	39
leeftijd_vanaf	26
lesrooster	46
min	200
min_en_max/hooguit	9
opleiding	213
oud	146
ouder_niet_dan	6
schoolverlater	115
stap	100
start	402
stud	239
tot	342
tussen	171

Table 3.3: Overview of the keyword occurrences in the annotated dataset

3.2.3 Evaluation Measurements

Three different focus points were considered when evaluating the annotated samples: General observations, keyword-based analysis and context analysis.

The main focus of the analysis of the regexes was put on False Negative samples. We hypothesized that exploring this specific set of samples would help to understand what is causing the somewhat low recall of the regexes and further, identify possible solution strategies to decrease the number of missed discrimination occurrences.

General observation

As a first step, we focused on all False Negative samples (annotated as discriminating but not detected by the regexes). This meant to read all False Negative Samples and exploring why this particular sentence was not matched by any of the regular expressions.

Keyword Evaluation

During the keyword evaluation, both quantitative and qualitative measures were applied.

First, it was compared if some keywords are more likely to occur in the group of False Negative samples than others. This was evaluated using the χ^2 goodness of fit test for every keyword, i.e., we used the χ^2 goodness of fit test to see if the observed frequency of a keyword in the False Negative Group was higher than the expected frequency based on the presence of the keyword in the annotated dataset.

As with the general observations, we additionally explored the annotated sentences on a keyword level. However, for this keyword-based analysis we focussed on all samples of a keyword and not only on the False Negative samples and created a list of properties we identified.

Context Analysis

Lastly, we conducted a context analysis, where we focussed not on the keyword but on the context of a keyword. Context means here the 5-grams to the left and the right of the keyword. The goal of this evaluation was to see how much the context of discriminating and non-discriminating samples differs. Therefore, this last evaluation purely focused on the labels assigned by the annotators.

We selected the keyword “jong” for this analysis, as it had a near clear split between discriminating and nondiscrimination samples. This was done with a qualitative and a quantitative approach. For the qualitative approach, we generated word clouds for both the 5-gram contexts for discriminating sentences and the 5-gram contexts for nondiscrimination sentences. Based on these word clouds, we tried to determine if any contextual differences between the usages of the word “jong” could be identified in the word clouds. Additionally, we evaluated if most of the difference in context is contained close to the keyword itself or within the whole sentence. This information would help to determine how much of the context a regex needs to cover to detect the discrimination. To compare the differences in word frequencies in the 5-gram context versus the whole sentence without the keyword, we used the Jensen Shannon divergence (JSD). The Jensen Shannon divergence (JSD) is a measurement which allows to evaluate the similarity between two probability distributions, in this case the distribution of words.

3.3 Findings

3.3.1 Annotator agreement

It was not possible to calculate the inter-annotator agreement between each annotator, since not every annotator had seen all sentences. We did, however, calculate the inter-annotator agreement between the two annotator groups using Cohen’s Kappa [15]. The annotator groups reached a substantial agreement of $\kappa = 0.61$.

	Discriminating	Non-Discriminating
Discriminating	1334	256
Non-Discriminating	6	1611

Table 3.4: Confusion Matrix Baseline

3.3.2 General Observations

Table 3.4 shows the confusion matrix for the regular expressions on the annotated dataset. Of all samples which were used for the evaluation, 1611 samples are *True Negative* (no regex match and annotators rated it as non-discriminating), 1334 *True Positive* (correctly identified discrimination by regex). 256 were *False Negative* (not detected discrimination), 36 samples were *False Positive* (actually non-discriminating marked as discriminating through regex). The matches were obtained using the flexible version of the baseline. 28 samples were assigned the tag “Other”, 21 “Internship”. That means that more actually discriminating samples were missed by the regexes (False Negative) than non-discriminating samples that were accidentally matched by a discrimination regex (False Positive).

An inspection of the FN samples highlighted three main issues with the regexes: *varying formulations*, *missing formulations and writing styles*, and *keyword coverage*.

Varying formulations and sentence structures

On the one hand, some sentences have an elaborate and nested formulations. The regexes are too static to detect the age discrimination if it is spread out across a long sentence.

We zoeken mannen, maar vooral ook vrouwen, die de werkvloer door en door kennen, en tussen de 50 en 70 jaar oud zijn.

On the other hand, some sentences use only a few words, possibly coming from a bullet point list of requirements, to describe the ideal candidate. For those samples, the regexes are much too detailed as they assume a proper sentence structure

-Leeftijd tot 27 jaar

Missing Formulations

For some sentences, the age discrimination is obvious and one of the current regexes can be easily tweaked to match this particular case as well. To give an example, by adjusting regexes for “stap” to also consider “derde stap”, a sentence like the following would be detected:

We zijn op zoek naar een kandidaat die de tweede of derde stap in zijn/ haar loopbaan zoekt

Further, there are also samples which do not match any regex pattern, but still clearly contain age discrimination.

<Company Name> streeft ernaar om jonge professionals tijdens hun studie kennis te laten maken met het succesvol toepassen van technologische ontwikkelingen.

Keyword coverage

Of all those false negative samples, there is no sentence which does not contain any keyword. I.e. there is no sentence which is discriminating but does not contain any of the keywords the regexes try to address. However, it needs to be kept in mind that the annotation sample set laid a focus on words containing a keyword. Only a total of 297 samples in the original dataset generated for annotation did not match a

keyword. We did, however, observe that those false negative samples with keywords still revealed new potential keywords, which the regexes do not account for. This is highlighted by the following sample:

heb jij je diploma onlangs gehaald en wil je aankomend jaar een centje bijverdienen?

This sentence contains a keyword (*bijverdienen*), but the phrase *diploma onlangs gehaald* actually entails the age discrimination. The set of samples without keywords was chosen to be small, as it was not the main focus of investigation. Therefore, considering also the findings of Fokkens et al. [11, 1] it is likely that there are more keywords than the ones detected so far.

3.3.3 Keyword based analysis

keyword	TP	FP	TN	FN
<no keyword>	0	0	232	1
baan	94	0	81	5
babyboom	0	0	2	0
bijbaan	89	0	43	8
bijverdienen	9	0	18	7
ervaring	86	4	162	11
geboortejaar	0	0	2	0
generatie	0	0	48	0
jong	47	6	52	14
jong deel uit maken	1	0	11	2
jong groep passen	3	0	10	3
jong groep versterken	0	0	0	1
leeftijd	1	0	72	16
leeftijd bepaald	0	0	28	0
leeftijd tot	0	1	30	4
leeftijd van tot	1	0	14	23
leeftijd vanaf	2	7	4	13
lesrooster	0	0	43	3
min	95	1	89	14
min en max/hooguit	0	0	9	0
opleiding	105	5	97	4
oud	50	4	85	6
ouder niet dan	2	0	4	0
schoolverlater	75	0	3	36
stap	76	0	21	2
start	143	3	64	14
starte	134	2	29	13
stud	161	1	48	20
tot	77	3	228	32
tussen	83	1	82	4

Table 3.5: Overview of confusion scores for each keyword

Table 3.5 shows the total numbers of True Positive, True Negative, False Positive and False Negative (i.e., confusion scores) for each keyword. This table is taking duplicates into account, i.e., it counts a sentence which contains two keywords into the results for each keyword.

To see if certain keywords are more likely to be missed by the regexes, the distribution of a keyword within the annotated dataset and within the False Negative group was calculated. Using the H_0 hypothesis of “The frequency of a keyword is the same in the annotated dataset and the group of False

Keyword	χ^2	Change in Frequency
opleiding	0.154	↓
tussen	0.0847	↓
schoolverlater	0.079	↑
leeftijd van tot	0.0677	↑
baan	0.0654	↓
stap	0.0657	↓
leeftijd vanaf	0.0362	↑
ervaring	0.0334	↓
leeftijd	0.0197	↑
oud	0.0188	↓
bijverdienen	0.0106	↑
bijbaan	0.0043	↓
jong	0.035	↑
tot	0.0035	↑
jong groep versterken	0.0033	↑
start	0.0033	↓
leeftijd tot	0.0016	↓
jong deel uitmaken	0.0009	↑

Table 3.6: χ^2 goodness of fit results for each keyword

The χ^2 goodness of fit was done for each word individually. The horizontal break in the table marks the critical value of 0.004. For all keywords above the line, the H_0 hypothesis needs to be rejected, for all below not. The last column indicates if the frequency is lower (↓) or higher (↑) in the False Negative group than in the annotated data set

Negative samples”, a χ^2 goodness of fit test was conducted for every single keyword. The frequency within the annotated set was used as the expected frequency, the frequency in the False Negative group as the observed. With a p-value of 0.05 and the degree of freedom being 1, the critical χ^2 -value is 0.004.

The results are summarized in Table 3.6. It shows that sentences with certain keywords are more likely to be missed by regexes, the most prominent being “schoolverlater”. And other keywords appear less often than expected in the False Negative group, given their frequency in the whole annotated corpus. The drawbacks of the regexes seem to be affecting the majority of the keyword-categories. Six keywords (*babyboom*, *ouder niet dan*, *min en max/ hooguit*, *geboortejaar*, *generatie* and *leeftijd bepaald*) do not have any False Negatives and are therefore not represented in Table 3.6. All except *ouder niet dan* do also not match any regex.

When looking at False Negative samples sorted by their contained keyword, we could identify that certain keywords suffer more from specific issues. The focus on sentences from one keyword also revealed additional issues which caused samples to be missed by the regexes. Below, we first revisit the issues of *varying formulations and sentence structures* and *missing formulations* on a keyword level and afterwards extend the list of issues with additional observations from the keyword-based analysis.

Varying formulations and sentence structures

The previously identified issue of varying formulations and sentence structure seems to mostly concern the keywords “leeftijd” and “ervaring”. For these groups, bullet point phrases are more common in the False Negative group than for other keywords. However, the “leeftijd” False Negative samples also contain samples with formulations which are too elaborate to be captured by the regexes.

Missing formulations

For some keywords (e.g. *tussen*, *stap*), adjusting the regexes would yield total coverage for the annotation set. Additionally to the already identified new formulation of “jonge professionals” in the last Section,

we also discovered the formulation “werken met leeftijdsgenoten”.

High Context dependency

For the keywords “jong” and “schoolverlater”, the discrimination is set by the context in which the sentences are used which clearly implies that the applicant is expected to be either young or a recent graduate. The formulation are, however, too open. There seems to be no frequent formulation which can be added to the regexes to cover the samples in the False Negative group.

This particular observation will be explored in more detail in Section 3.3.4

Limited non-discriminating usage

When looking at the ways the word “schoolverlater” can be used in a non-discriminating way, we observed that only 3 samples are actually used in a non-discriminating way.

Keyword co-occurrence

Some keywords seem to cooccur with specific keywords. One example is the combination of *jong* and *leeftijd*. This combination is rather interesting because it seems to inherit the high context dependency of the keyword *jong*.

Further keywords which are frequently accompanied by other keywords are *bijbaan* and its different formulations (*bijverdienen*, *baan*). But as shown with the example of *heb jij je diploma onlangs gehaald en wil je aankomend jaar een centje bijverdienen?*, there are other words which “carry” the discrimination. The word *bijbaan* itself seems to be more of an discrimination-aiding keyword than a discrimination-carrying keyword. This highlights once more the importance of the context in which certain keywords are used.

3.3.4 Context Analysis

In the previous keyword analysis, we identified the keywords of *schoolverlater* and *jong* as the samples that are most affected by complex context dependency. Of those two keywords, “jong” has a balanced distribution in terms of discriminating and non-discriminating samples in the annotated dataset (58 sentences each). We therefore chose samples with the keyword “jong” for a more detailed context-analysis.

Word Cloud Observations

The word clouds for discriminating and non-discriminating context of the keyword “jong” confirmed the observations we made when looking at the sentences in terms of context dependency. The word cloud showing the discriminating context (Figure 3.1) highlights that “jong” in discriminating sentences is used to find young applicants for a job (“zoeken”, “voorkeur”, “medewerker”). But also several adjectives are often accompanied by the word “jong”, such as “enthousiaste”, “ambitieuze”, “spontane”. This insight might provide some interesting starting points for inclusive tone-of-voice research. Looking at the non-discriminating context word cloud, we see that the word “jong” can also be used to describe the job itself. In those cases, the usage is not discriminating, as it refers to “kinderen”, “bedrijf”, or in general “mensen” the applicant will have to work with.

However, comparing both word clouds, we also find an overlap of certain words and phrases. The word clouds (Figure 3.1 and 3.2) show that the word “jong” and “leeftijd” is a common combination for both discriminating and non-discriminating sentences.

Context Similarity

The JSD distance between the word frequency in discriminating sentences and word frequency in non-discriminating sentences was 0.71. For the word frequencies only focusing around 5 words to the left and right of *jong*, the JSD was 0.73. This indicates that the direct surrounding of the words holds slightly more information as to which context the word *jong* is used in. But if one also considers the findings from the keyword-based analysis, focusing too much on the direct surroundings of a keyword can lead to missing elaborate formulations.

In general, the observations from the first inspection of the False Negative group were confirmed and also extended when looking at the keywords. But more importantly, it could be shown that certain issues seem to affect certain keywords in particular.

3.4 Discussion

This study aimed to highlight the strength and weaknesses of the current Dutch Baseline for age discrimination.

3.4.1 Strength

Even though the analysis was mainly focussed on the False Negative samples, we observed only a few False Positive samples. This is in line with the reported precision by the original study and highlights the main strength of the regexes. Samples which are detected to be age discriminating are in the majority of cases truly age discriminating. Furthermore, even with a sampling targeted towards obtaining False Negatives, there were only 256 False Negative samples. Even though there is still some room for improvement in terms of recall, the regular expressions already cover a big part of the possible phrasing of age discrimination in job advertisements.

3.4.2 Weaknesses

Our evaluation of the False Negative samples revealed several weaknesses, which mainly arise through the rigidity of the regexes. Regarding the issues of *varying formulations and sentence structures*, our results showed that regexes are both too wide while also being too narrow in their formulation. This highlights that even the flexible version of the regexes is still not flexible enough to account for the complexity of natural language. The issue of *missing formulation* is yet another indication that the regexes struggle to deal with the complexity of natural language, especially with the vast amount of possible formulations for the same message. For the keywords “*jong*” and “*schoolverlater*”, we observed that the discrimination is dependent on the context the word is used in. Here it becomes apparent once again that one regular expression can only capture one small aspect of the context, and that it needs an elaborate list of expressions to capture every possible formulation. Lastly, we observed that some keywords have a limited observation usage and others tend to co-occur. One additional drawback of the regexes is the focus on the lexical aspects only. The regex list from the baseline only tries to model age discriminating phrases and does not exploit the fact that some keywords are maybe more like to be discriminating than non-discriminating. This could have been done through modeling negative regular expression, which model non-discriminating usage rather than discriminating usage of certain keywords. Further, regular expression only focus on the wording of the sentence and do not take any higher level features such as, e.g., part of speech into account.

3.4.3 Implications for a Hybrid Approach

The observations we listed above show that the regular expressions provide a good baseline for age discrimination detection. One of the main benefits next to the high precision which should be maintained by any hybrid approach is the explainability of the regexes. All regex-based decisions can be fully

understood, and it is possible to intervene and adjust individual regexes if needed. Machine Learning techniques on the other side are known to be much less explainable and more of a “black box” approach. Therefore, regular expressions could be a good final decider on the discrimination detection label.

But given the weaknesses we detected above, any hybrid approach would need to address the incapability of regexes to deal with the complexity of language. A solution to this would be a much more elaborative list of regexes to account for all the possible formulations age discrimination can be conveyed in. But crafting regexes is time-consuming, laborious, and prone to errors. Furthermore, there are so many formulations to be considered that this seems like an unworkable option. Any way to support this process through Machine Learning Techniques would benefit the detection of age discrimination while maintaining explainability and precision of the regexes.

Regarding what the regex should aim to capture, our findings showed that, at least in the keyword “jong”, there is more information in the direct surrounding of the keywords. The current list of regexes focuses exactly on this already and given that already this restricted focus ended up in missing formulations, keeping this narrow focus on the keyword seems more promising.

3.4.4 Limitations

The first limitation concerns the detection of age discrimination in general. Even though objective age discrimination is defined through a legal framework, it can be hard to decide whether a sentence is age discriminating. This is due to age discrimination being a special case of discrimination which is particularly hard to detect [7]. As stated before, discrimination also depends on the person being addressed and reading the job description [6]. This creates another limitation for the annotated data, since all annotators are not member of the discrimination target group. It is possible that their age caused the annotators to miss discriminating samples which persons of the target group would have rated as objectively age discriminating. Further, none of the annotators is an (age) discrimination expert. Most of their domain knowledge was obtained through the companies recent focus on age detection discrimination and the annotation guide.

An additional limitation concerns the samples itself. Annotators reported that in some cases, one sentence alone was not enough to decide on a label. Furthermore, some sentences seemed to have been additions to a job description to encourage certain age groups to apply which were not targeted with this job description. E.g., a sentence could say, “We also highly encourage starters to apply to this posting.”. Without the full context of the job description, it is hard to decide whether this would discriminate older people, as they might have been targeted in a similar sentence at another place in the job description.

These limitations can have an impact on the findings of this analysis; however, we had to make these decisions due to economic reasons.

3.5 Conclusion

We found that regexes are a good baseline to detect age discrimination, as they have a high precision. But with an eye towards recall, we observed that they struggle with the complexity of language due to their rigidity. Considering the overall research goal of this thesis, we also evaluated the role of regexes in a hybrid approach, highlighting aspects Machine Learning could improve.

Chapter 4

Language Model

To explore in which way Machine Learning techniques can be applied to improve the detection of objective age discrimination, we conducted a visual vector space evaluation. Section 4.1 gives an overview of the used approach, followed by Methods and Findings. We conclude the chapter with a discussion of these findings.

4.1 Background

When using natural language as an input to machine learning approaches, a good representation of the word is required. The common representation uses a technique called *word embeddings*, which represent a word as a vector in a vector space.

4.1.1 Word Embeddings

Word Embeddings are usually obtained by extracting the weights of the last layer of a network which is trained on a specific corpus of text documents. If trained correctly, the embeddings capture the lexical context of the words from this corpus. In practice, this means that words which often occur in the same context will have similar vector representation and therefore be located in proximity in the vectors space. This property is particularly useful since it has been shown that similar words often occur in the same context, and therefore well-trained word embeddings can not only capture lexical similarity but also semantic similarity [16, p.783].

Word2Vec

The first approach to obtain high-quality word embeddings has been proposed by Mikolov et al. [17] and is known as the *Word2Vec* model. It is also the approach used by Burn et al. to compute the similarity of job descriptions to ageist stereotypes [12]. One of the main limitations of the Word2Vec approach is that it is not fit to handle homonyms (words which have more than one meaning). The model cannot differentiate if, e.g., the word “bank” is meant to be the financial bank or the river bank. This makes the embeddings of homonyms words imprecise, as it tries to capture two or more mental concepts in one vector. Contextualized word embeddings address this problem.

Contextualized Word Embeddings

Since the Word2Vec model came up the first time, several studies have improved the state of the art of word embeddings by focusing on more complex representations and contextualization of the former to address polysemy among other things (e.g., [18] and [19]). Contextual word embeddings do not only encode the word, but also try to capture the context the word is used in within the representation. Instead

of treating the context of the word as a bag-of-words, models for contextualized word embeddings take the direct context of the words into account. This means that a word has a slightly different embeddings for each different context and always has to be computed using the whole sentence as input.

4.1.2 BERT

With the introduction of the transformer model in 2017, the state of the art in Natural Language Processing has been boosted and the current best known word embedding model is the BERT model [20]. BERT [21] stands for Bidirectional Encoder Representations from Transformers and is the first model that is considered to be truly bidirectional. Bidirectionality of word embeddings means that both the left and the right context of the word is encoded in the word embedding. Prior approaches usually trained their word embeddings by using the sentences both in a forwards and backwards fashion, and created a combined word embedding from the outputs of both models. BERT approaches the problem of bidirectionality from an entirely new angle and utilizes the Cloze learning task, in which the element to be learned in context is masked, and the learner has to name the masked word. In the following, we will give a brief explanation of the BERT architecture. Like the original BERT paper, we refer to the original Transformers paper [22] for a detailed explanation of the transformers model, the underlying model of the BERT architecture.

The BERT framework

The BERT framework [21] consists of two steps: pre-training and fine-tuning. For the pre-training phase, the model is trained on an unlabeled corpus to learn the representation of the words (the word embeddings). This phase consists of two learning tasks. The first is the already mentioned Cloze or Masked Learning task for which 15% of the input tokens are masked (replaced by a mask token) and the model has to predict the hidden word. As the second learning task, BERT has to predict if the second sentence in a sequence of two input sentences is actually the next sentence. This task is specifically useful if the BERT embeddings are later used in Question-Answering tasks. In those cases, the answer would be the second sentence following the question and the BERT prediction indicates if the answer is the correct one given the question. For non-question answering approaches, it was shown that the Next Sentence Prediction (NSP) task can be omitted without harming the performance.

Once the pre-training phase is finished, the model can be fine-tuned. For this, an additional layer is added to the output of the pretrained BERT model and, using labeled data, the weights are tuned to solve the specific problem at hand. The benefits of the split in pre-training and fine-tuning, an approach which is known as transfer learning, is that the base model can be trained on a general dataset (e.g., the whole of Wikipedia). Downstream application then only have to fine-tune the already pretrained BERT model to their specific task.

4.1.3 Albert

Various BERT like models have been proposed since the publication of the original BERT paper. One of the most recent ones provides a lighter version of **BERT**, ALBERT [23]. By using parameter reduction techniques, it was possible to reduce the amount of required parameters for BERT (from 110 million to 12 million, 18 times fewer than before) which allows training 1.7 times faster than before without hurting the performance. In fact, ALBERT sets a new state-of-the-art for several benchmarks (GLUE, RACE, SQuAD).

4.2 Methods

4.2.1 Data

To train BERT, we used the *2014 job description Dataset*, introduced in Chapter 2. BERT like models work on sentence-level, which required the dataset to be split into sentences. We described this process

in the before mentioned chapter. The annotated dataset (see last chapter) was used afterwards as an input to the language model. The resulting word vectors formed the basis for the visual analysis.

4.2.2 Training BERT

In most cases, when used for downstream applications, BERT models are only fine-tuned to the specific task rather than trained from scratch. For this study, we are, however, interested in the word embeddings as such to see if the latent space spanned by these word embeddings captures objective age discrimination to some extent.

Tshitoyan et al. [24] showed that for a material science-based vector space analysis that a smaller but domain relevant data set captures semantic relationships better than a bigger but domain-unrelated dataset (e.g., Wikipedia articles). This goes against the common assumption that more data is always better, and motivated our choice to train BERT from scratch for our analysis.

For the pre-training of our ALBERT model, we made use of the huggingface library [25], which provides an implementation for both the ALBERT model and the SentencePiece tokenizer which is needed to process the input corpus. Our input corpus is the *2014 job description dataset*. We trained ALBERT for 1 epoch with a batch size of 32 and choose the default 15% masking rate for the masked learning task. We only used the masked learning task and did not add the next sentence prediction task to the learning task for the training.

4.2.3 Obtaining the word embeddings

We use our pretrained BERT model and extract the last hidden layer as word embeddings. For this, the BERT model receives one sentences as input. The output is a list of 768-dimensional vectors. These tokens are padded in between two additional vectors, starting with the [CLS] classifier token and ending with the [SEP] which marks the end of the first input sentence. To obtain a representation for the sentence, we followed common praxis and averaged over the embeddings for the sentence tokens [20]. Eventually, each sentence from the annotated dataset was represented by a 768-dimensional vector.

4.2.4 Visualizing the word embeddings

To be able to visualize the vector space spanned by these embeddings, it is necessary to reduce their dimensions. One of the most common reduction approaches, especially for word embeddings, is the so called t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [26]. T-SNE is a non-linear approach to dimensionality reduction, which tries to maintain the high dimensional neighborhood of a data sample in the low dimensional space [26]. One important tunable parameter for t-SNE is the perplexity, which allows the user to define how the reduction should balance local and global structures within the data [27]. As a rule of thumb, the perplexity should usually not be bigger than the number of samples provided. A good way to find the correct perplexity is by considering it as an estimate of how big the direct neighborhood of a sample is [27]. The default range given in the original paper [26] is between 5 and 50, but also higher values can be appropriate [27]. It is advised to plot several configurations to choose the correct value. Following this advice, we reduced the dimensionality with a range of perplexity values (between 20 and 100 in 5-step increments).

We generated three different types of t-SNE plots. The first t-SNE plot displays all word embeddings, and we colored each embedding based on the keyword it contained. This was done as a sanity check to see if the word embeddings somewhat capture the keywords. The second t-SNE plots displays the same data points as the first, but here the colors represent discriminating/non-discriminating samples. The last t-SNE plot type was done for each keyword, also coloring the samples by their discrimination label.

Visual Exploration

Evaluating the quality of word embeddings is a hard task, especially when only little annotated data is available. To explore and evaluate the space the word embeddings form, also known as *latent space*,

domain knowledge and human judgment is required [28]. In our case, we derive this domain knowledge from labels of the annotated subset of the *2014 job description dataset*, which creation was described in the last Chapter. We approach this evaluation with the general question: *Does the latent space capture the objective age discrimination?*. To answer this question, we defined a list of three sub-questions:

- *Can we find clear clusters with correspond to the discrimination labels for the respective sentence?*
- *Do the word embeddings capture the specific context in which a keyword is used? I.e., do the embeddings for a certain keyword form clusters which correspond to the different contexts a keyword is used in?*
- *Can we identify issues and problems with the word embeddings, e.g., are there regularities in outliers?*

To answer these question, the researcher of this study conducted a manual visual exploration of the before described plots and formed generalized observations based on the found structures found in the plots.

4.3 Findings

We used the T-sne plot showing all samples of the annotated dataset colored by the contained keyword to choose the right perplexity (perplexity = 25) (see Figure 4.1). The generated plots for this perplexity can be found in the Appendix, and interactive versions are available at <https://tinyurl.com/42u3dc6p>.

4.3.1 General View

Figure 4.2 and 4.1 show the t-SNE plot for all annotated data samples. In Figure 4.2, we colored each sample based on their discrimination label. Even though there appears to be a region on the lower left part of the plot which is nearly exclusively non-discriminating, we cannot identify clear discriminating/non-discriminating clusters. However, this is not too surprising as the annotation was made based on the Dutch legal framework.

4.3.2 Keyword-Based Analysis

After inspecting the labels from an overall perspective, we continue our exploration with the keyword plots. This led to several observations.

There is a general tendency of context clusters

For most of the keywords, there is at least a tendency of samples to group based on their context. Here, the most prominent contexts are usually *“Looking for a candidate”*, *“Description of the job”*, *“Advertising specific type of job”*, *“Salary”* and in some cases *“Age requirements”*.

Table 4.1 shows sample sentences for each of these contexts. Generally, sentences addressing the candidate in the second persons are grouped together and distinct from samples describing the job. Only when the candidate is described in the third person, it might be located within a cluster of job describing samples.

Only inherently discriminating context are purely discriminating clusters

If these contexts are inherently discriminating, e.g. *“Age requirements”*, we observe a pure cluster of only discriminating samples. This can be seen in Figure 4.3, which visualizes the embeddings for all sentences that contain the keyword “min”. All samples which are located in the black square on the right-hand side are all dealing with age requirements of the applicants, which is an inherently discriminating context.

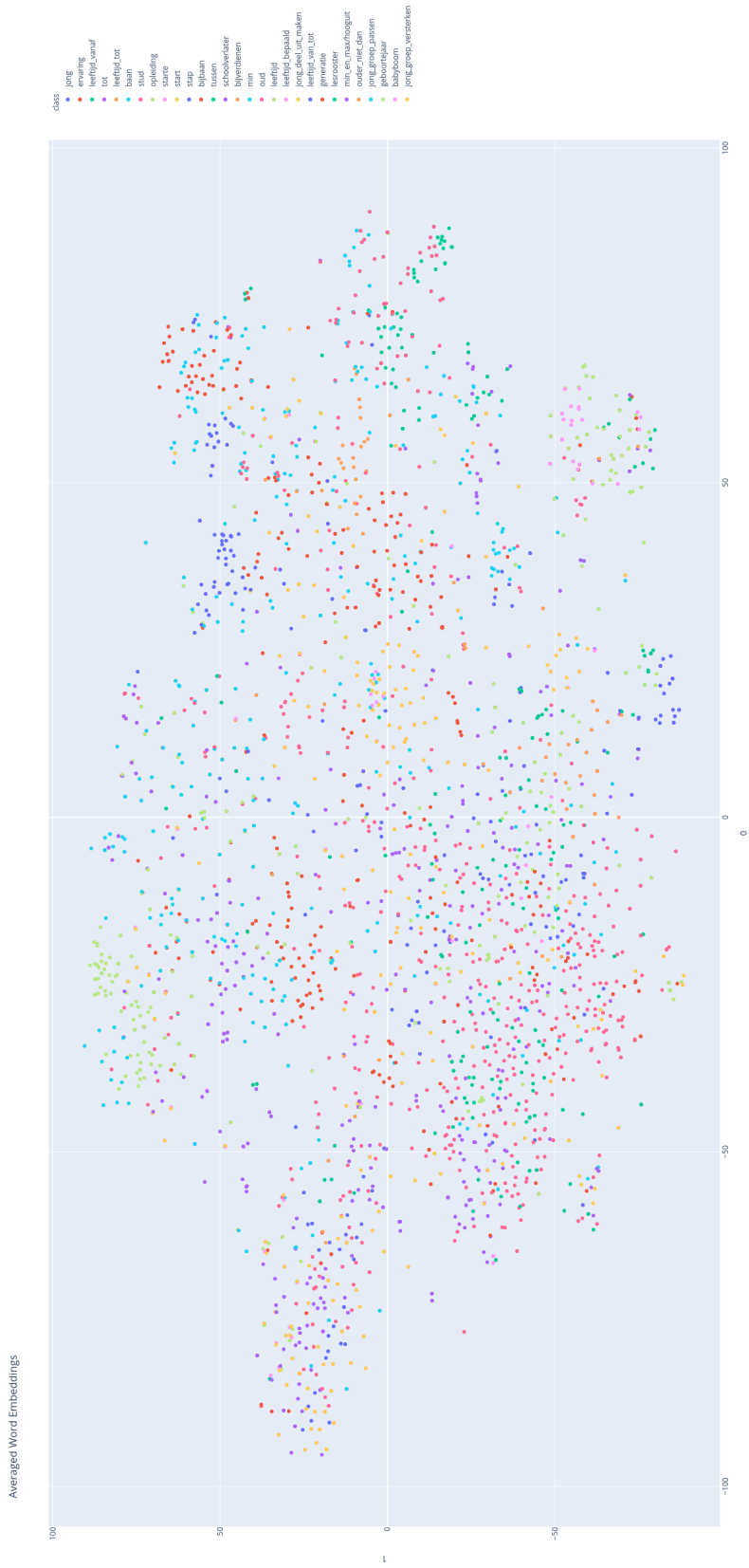


Figure 4.1: All reduced embeddings for the annotated subset of the 2014 job description dataset, keywords colored

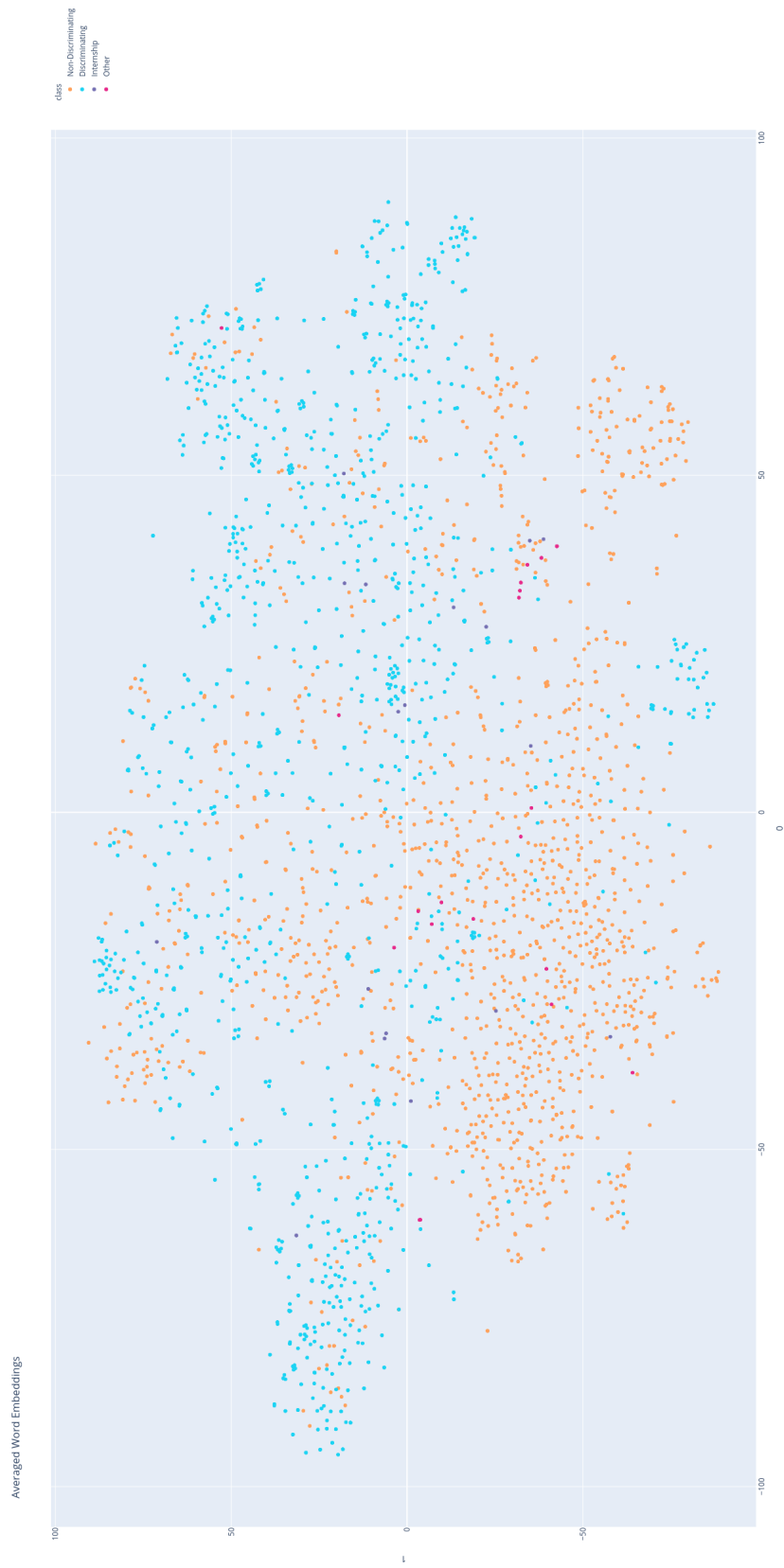


Figure 4.2: All reduced embeddings for the annotated subset of the 2014 job description dataset, discrimination labels colored

Context	Sample sentences
Looking for a candidate	“Voor verschillende gerenommeerde bedrijven in de Techniek zijn wij op zoek naar jong talent en ervaren elektromonteurs” “Voor onze showroom in Amsterdam zoeken wij een leuke jonge spontane enthousiaste collega”
Description of the job	“Het betrekken van jongeren en actieve bewoners (jong en oud) bij de wijk, buurt, straat waar ze wonen” “Naast oudere patiënten heb je ook te maken met jongeren met een chronische aandoening”
Advertising specific type of job	“Interessante afwisselende baan” “Een leuke en uitdagende baan voor een starter op de arbeidsmarkt”
Salary	“Salariëring afhankelijk van ervaring” “De functie is ingedeeld in functiegroep 4/5, het salaris wordt afgestemd op kennis en ervaring”
Age Requirements	“Minimaal 21 jaar oud” “Ben je minimaal 23 jaar oud”

Table 4.1: Overview of common contexts that form clusters with sample sentences

Figure 4.4 visualizes the word embeddings for all sentences that contain the keyword “ervaring”. The lower right-hand side of the plot shows a cluster of non-discriminating sentences. This cluster contains sentences that state that the expected salary of the candidate will be based on their experience. This is an inherently non-discriminating context, which shows that the observation we made before with discriminating samples can also be made with non-discriminating samples.

In contrast, the dense area on the upper part of the plot gives an example of a not inherently discriminating context. The samples in this part of the plot is composed out of sentences which all deal with the required number of years of experience. It is not always discriminating to talk about years of experience, however, limiting the number of years (e.g. “max 3 years of experience”) is. The context of years of experiences is therefore not inherently discriminating which is also reflected by a dense region that contains discriminating and non-discriminating samples.

One example for this can be seen in the plots for the keyword *ervaring* in Figure 4.4. The upper part of Figure 4.4 shows a sample dense region of both discriminating and non-discriminating samples. Most samples in this region list certain types of experiences the applicant needs to have. The discriminating samples limit the experience time (e.g. “max 3 years of experience”). Here, the context of different required experiences is not inherently discriminating, which is also mirrored in the composition of the grouping in terms of discriminating and non-discriminating samples. On the lower-right side, however, we see a clear non-discriminating context containing sentences that state that salary depends on the amount of experience of a candidate.

The embeddings capture meaning of digits from a broad perspective to some extent

The non-discriminating samples below the frame in Figure 4.3 include the word “min” and digits too, but describe working hours. They are still somewhat separated from the rest of the cluster, but they do not form their own distinct cluster. However, this could also be due to the small number of samples, since only four samples are non-discriminating. The keyword “tussen” presents a similar case with numbers in discriminating context and non-discriminating context but with a more balanced distribution of samples. Figure 4.5 shows that for “tussen” the two groupings are more separated, and it appears like the embeddings did capture the difference in the meaning of the digits.

Looking at a combined plot of both “tussen” and “min” (Figure 4.6 and Figure 4.7), it becomes clear that the embeddings capture the difference between the digits in a working hour context and the digits in an age requirement to some extent. In fact, the discriminating context was captured better than the

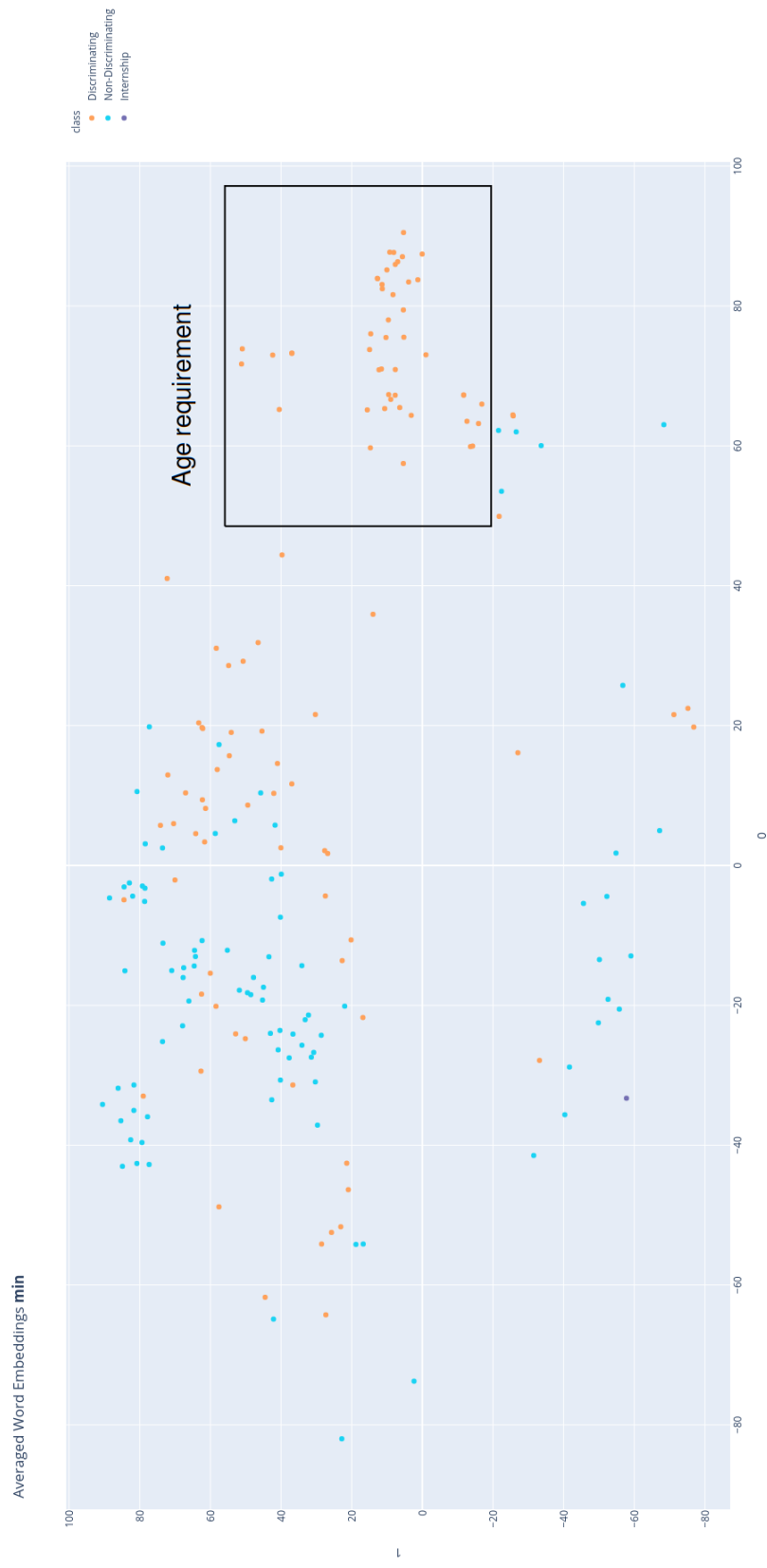


Figure 4.3: All reduced embeddings for the annotated subset of the 2014 job description dataset that contain the word “min”; discrimination labels colored

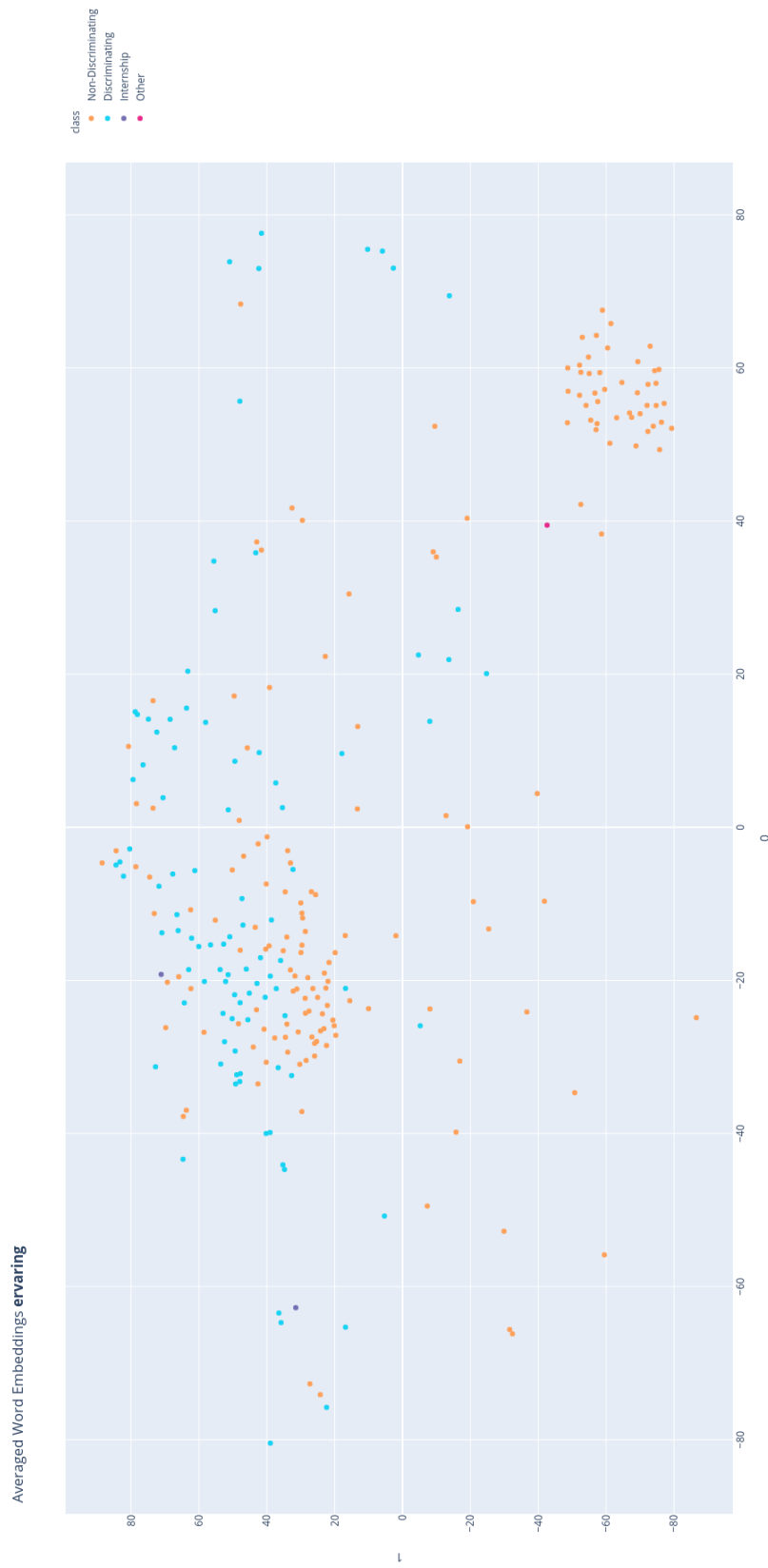


Figure 4.4: All reduced embeddings for the annotated subset of the 2014 job description dataset, keywords colored

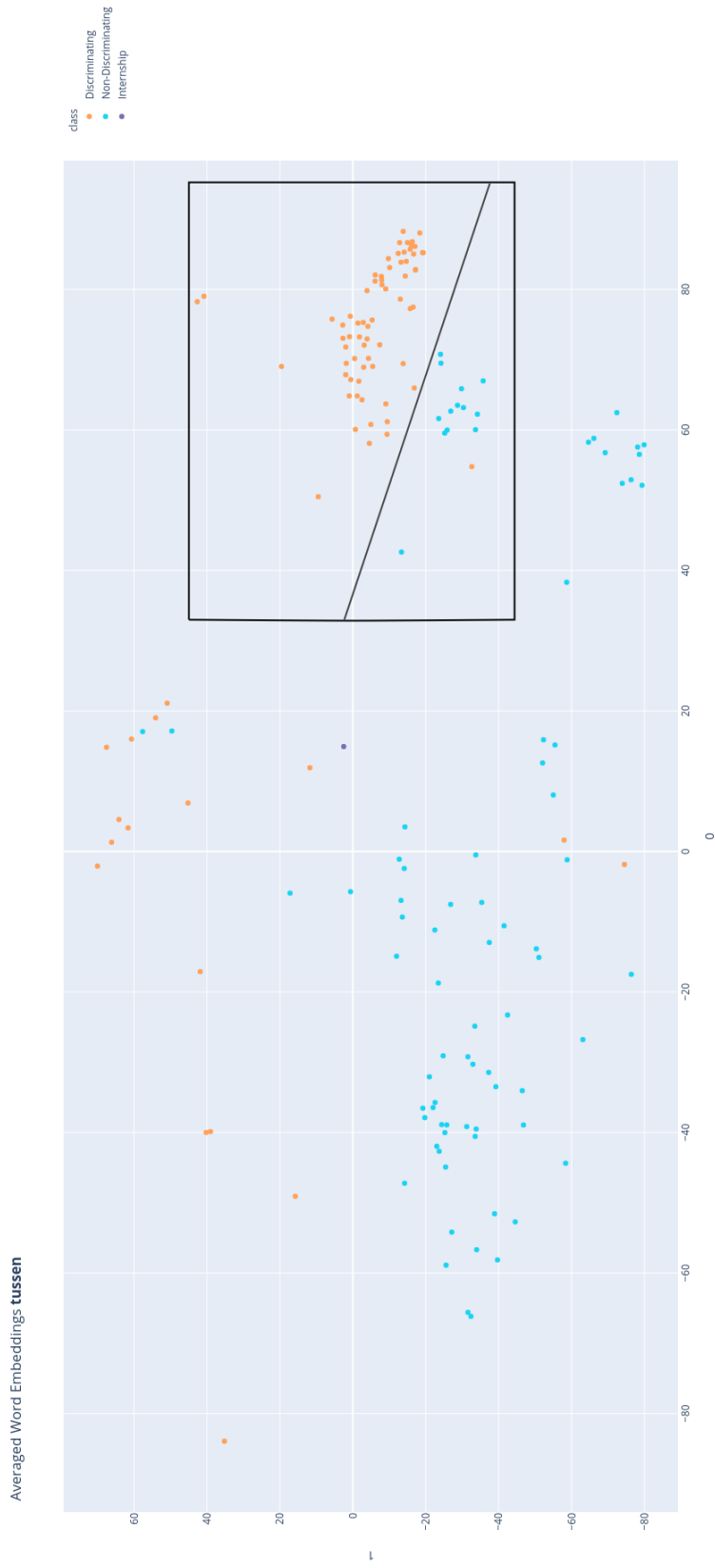


Figure 4.5: All reduced embeddings for the annotated subset of the 2014 job description dataset, keyword: “tussen”, colored by discrimination label

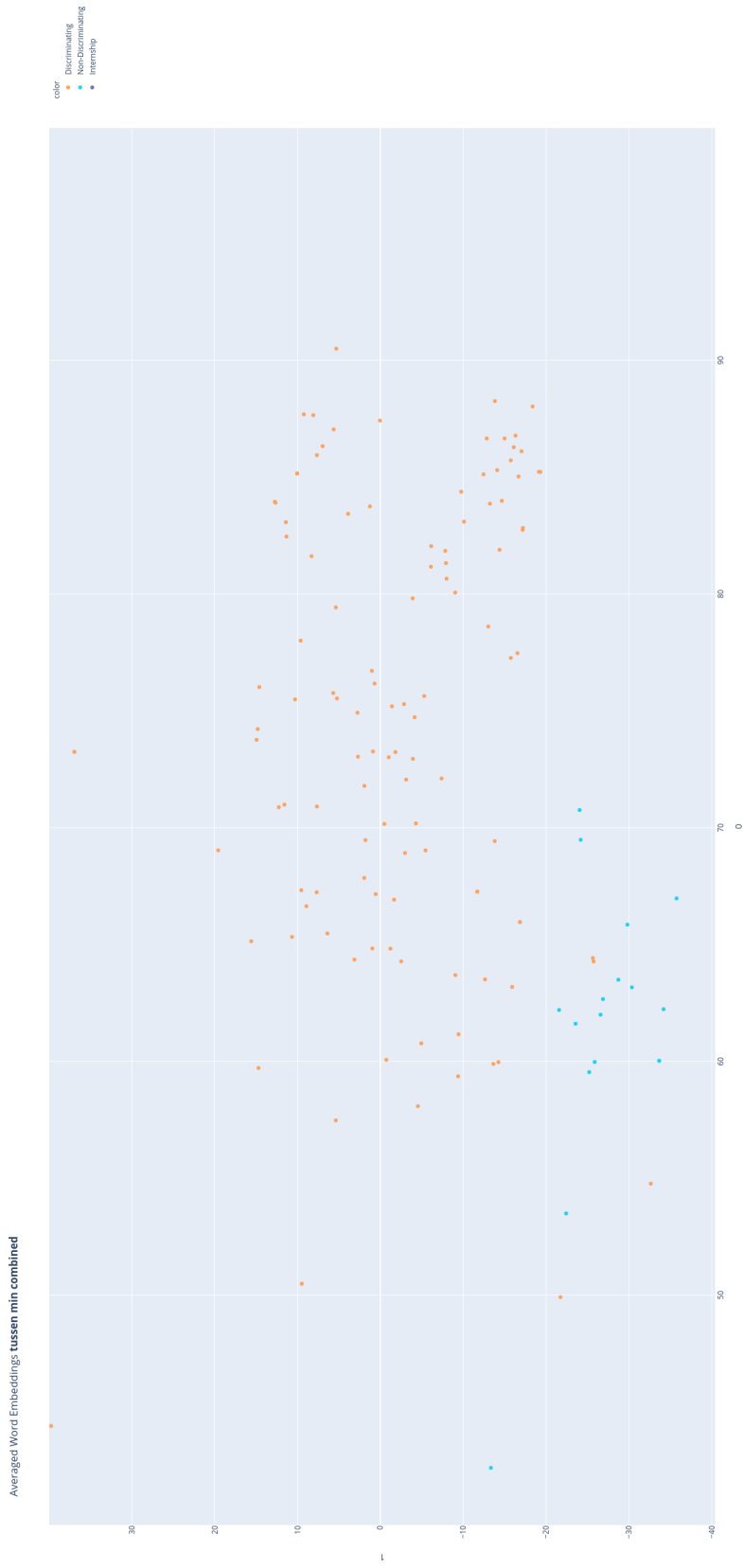


Figure 4.6: All reduced embeddings for the annotated subset of the 2014 job description dataset, keywords: “tussen” and “min”, colored by discrimination label

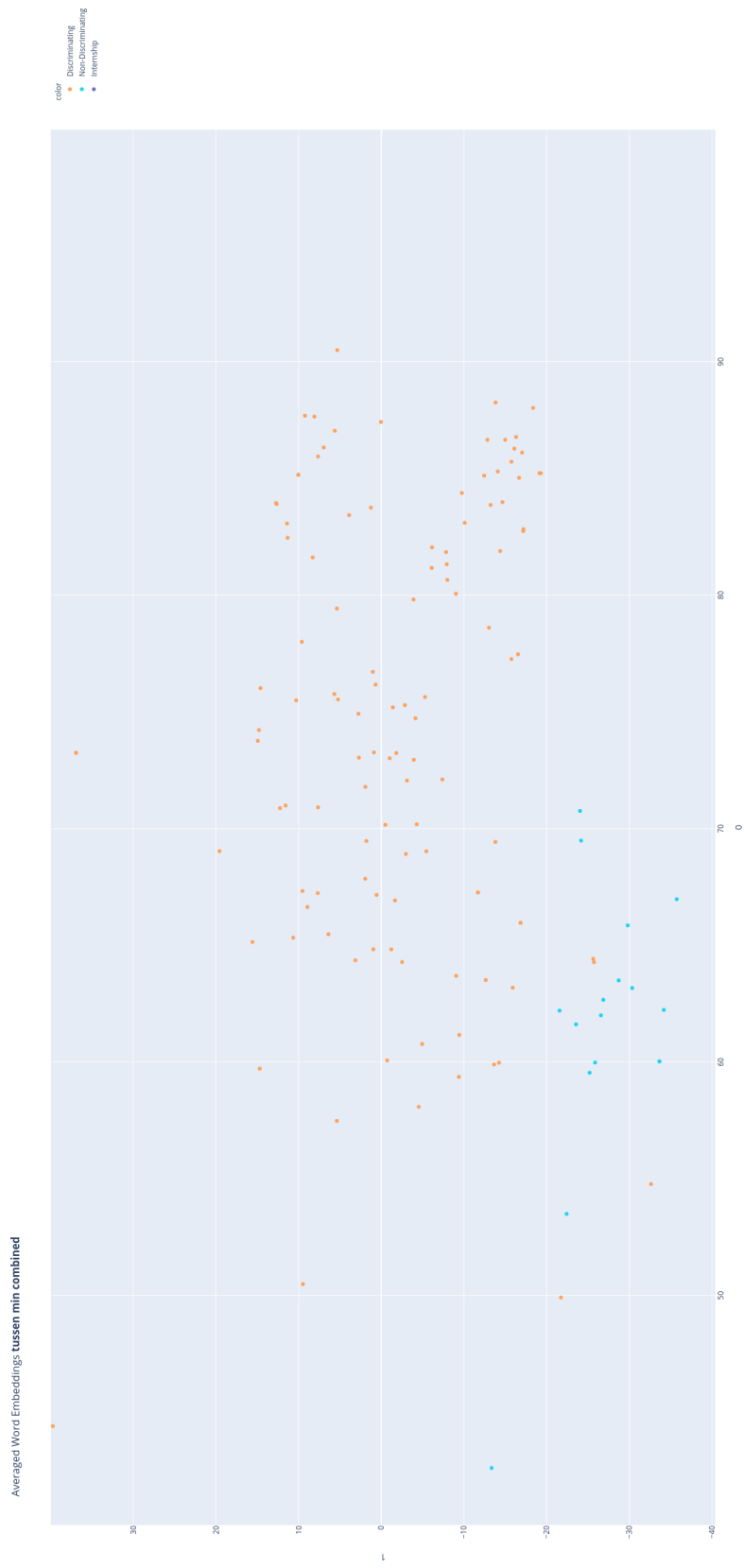


Figure 4.7: All reduced embeddings for the annotated subset of the 2014 job description dataset, keywords: “tussen” and “min”, colored by keyword

keyword, as it can be seen by comparing the coloring of Figure 4.6 and Figure 4.7. The samples are not exactly split into two clusters, but a noticeable separation between discriminating and non-discriminating samples beyond the keywords can be observed.

Sentence length matters

Looking again at the region within the square of Figure 4.3, we noticed two distinct areas of discriminating samples. The context here is the same, but the samples on the top are more elaborated sentences and the samples below more in bullet point style. We also observed this behavior in other plots, where longer sentences were often a bit of distance or at the edge of their clusters.

Extremely long sentences are isolated in plots

In line with the last observation is the observation that samples with a lot of text (either caused by failed sentence splitting or sentences which are a list of requirements) are usually isolated in the plots and do not appear in any dense layer region. This also holds for sentences which contain two or more concepts, which could be explained by the fact that the embeddings of this sentence is somewhat located in the middle of both concepts.

The averaging may cause problems if the wording order matters

Lastly, we observed that in some cases, purely averaging over all word in a sentence may cause sentences to be misinterpreted. One example for this can be found in the plot for *jong*. There is a somewhat distinct cluster with sentences expressing that a company is looking for young employees. However, one odd sample says that a company is looking for an employee to look out for young plants. This can be seen in Figure 4.8

4.4 Discussion

The aim for of this study was to survey if latent space formed by word embeddings trained on the *2014 job description dataset* captures objective age discrimination. To answer this question, we defined a list of sub questions, which will be discussed in the following, based on the observations we made during the visual exploration.

4.4.1 Clustering based on discrimination labels

The visual exploration did not reveal a clear clustering which corresponds to the discrimination labels. This holds both from a general perspective (considering all keywords), nor for individual keywords. This observation is not too surprising. The legal framework for age discrimination is a quite abstract formulation which is already hard to be understood by humans. Further, even though it is the most occurring form of discrimination, it only occurs in around 3% of job advertisements and consist mostly of one or two sentences within those job advertisements. This makes the number of discriminating sentences in the training corpus extremely low and there are much more prevalent semantic concepts within the training corpus, which in turn are more likely to be captured by the word embeddings.

4.4.2 Context detection

We were further interested in how well the word embeddings can capture context. Our exploration revealed that there is a general context cluster tendency, and we were further able to somewhat map certain clusters to topics/ real-world contexts, e.g., “*Description of the job*”, “*Looking for a candidate*” etc.

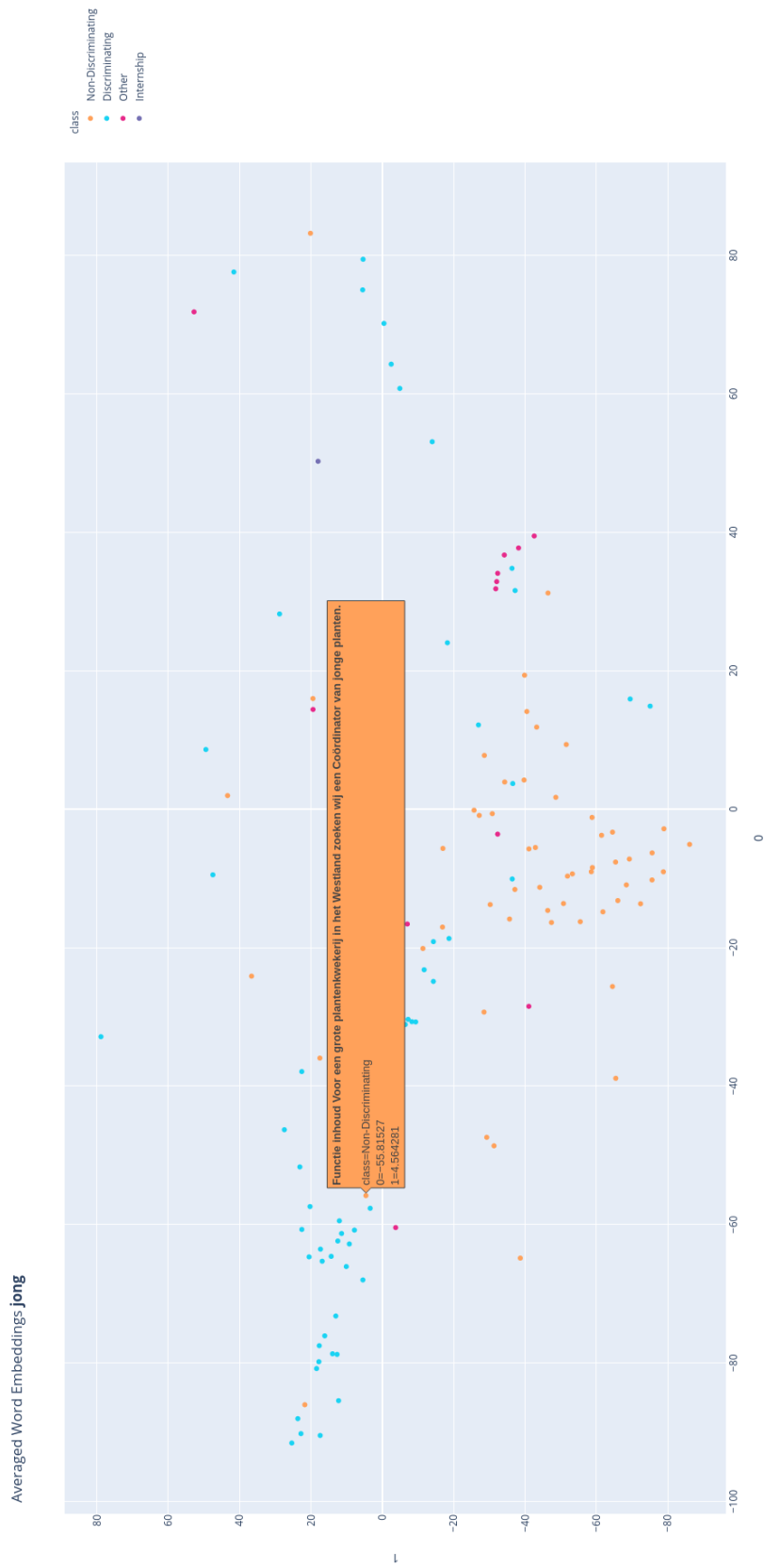


Figure 4.8: All reduced embeddings for the annotated subset of the 2014 job description dataset, keyword: jong, colored by discrimination label.

We also observed that certain contexts seemed to align with discrimination labels. But since this only happened for contexts of keywords which were inherently discriminating, we conclude that the context was captured here rather than the discrimination aspect.

4.4.3 Regularities in outliers

Lastly, we identified certain patterns in outliers and identified potential problems with the regexes. The findings suggest that the sentence length has a significant impact on the results, and it appears that the bag-of-word averaging may cause problems.

We observed that the embeddings of snippets (e.g., from bullet points) are somewhat separated from full sentences. A possible reason for this could be that the embeddings of bullet points have fewer words, which could “shift” the averaged embeddings in the direction of the additional words which are included in a sentence.

This could also explain why extremely long sentences tend to be outliers in the plots. They contain more words, which introduces additional semantic information to the sentence, resulting in a potential drift towards other “concepts”. The potential problem with averaging of the sentences is surprising. Since each embedding contains not only information about the word itself but also the context, averaging over all embeddings should maintain information about the order of words. This, however, happened only for one sample and may be only a rare case. In general, we could find no evidence that the averaging of the word embeddings caused any problems.

To summarize, even though we could find some clusters which coincide with the discrimination labels assigned by human annotators, we believe that our model captures the context of keywords rather than the discrimination labels. Therefore, we conclude that our model did not capture objective age discrimination. However, we made a list of interesting observations which may support the detection of age discriminating samples. We will discuss the implications for the development of an age discrimination detection approach utilizing word embeddings below.

4.4.4 Implications for a Hybrid Approach

As already discussed, the main advantage of regular expression over Machine Learning Techniques is their transparency and explainability. Word Embeddings capture information in dimensions which are incomprehensible for humans [29]. Therefore, they should not be responsible for the final decision about an age discrimination label in an explainable approach.

However, this higher dimensionality can capture information which is inaccessible to any human judgement. Machine Learning algorithms can handle a vast amount of data and detect patterns in it. On the one side, this means that it captures information beyond what humans can comprehend. But on the other hand, it also opens possibilities to detect patterns which would have been hidden from humans given the amount of job descriptions available.

Our visual analysis provided evidence that the model did capture certain contexts, which we were able to map back to real life concepts; even though we have no way of verifying how the model reached this conclusion. It becomes apparent that fine-tuning the ALBERT model would not yield in an explainable classifier and despite their shortcomings, regexes provide the better mechanics for the final prediction. However, the ability of word embeddings to capture contexts and sort data into somewhat meaningful groupings would allow to improve the discovery process of formulations of discriminations which can be used to improve the coverage of the regexes.

4.4.5 Limitations

As already mentioned, visual exploration of the vector space is a hard task. There are several limitations which apply to our results.

Firstly, to be able to visually explore the word embeddings, it was necessary to reduce the dimension. This reduction can potentially result in loss of information, and the 2-D representation of the word

embeddings is merely an approximation of the real latent space. Even though we carefully selected the proximity value, there are no tests which can be done to verify our selection.

Furthermore, we used a dataset which was built using specific keywords. Our results can only make statements about age discrimination contained in samples using these keywords.

Concerning our ALBERT model, it needs to be considered that we trained it on significantly fewer data and for fewer epochs than the original model by Lan et al. However, this decision was motivated based on findings from Tshitoyan et al. [24].

Lastly, the BERT-framework is commonly considered to be a black box. To this point, it is not fully known or understood why it works so well. We tried to use domain knowledge to interpret the result shown in the t-SNE plots; however, there is no way of knowing for sure if our interpretation and the connections we drew to the real world were the basis for the decisions of the ALBERT model.

4.5 Conclusion

This chapter presents our second sub-study. We explored if a language model trained on the *2014 job description dataset* captured objective age discrimination. We evaluated this through a visual exploration, which resulted in several observations, which informed a discussion about the place of word embeddings in a hybrid regex detection approach. Our findings suggest that objective age discrimination is not captured by the latent space. They do, however, suggest that the context of the keywords is captured. In cases where this context is inherently discriminating, we observed cluster formations that corresponded to discrimination labels.

Chapter 5

Regex Detection Tool

The last two chapters showed that a possible route to improved performance in the detection of age discrimination could be to use machine learning techniques to guide the discovery of new formulations which need to be addressed by regexes to increase their recall. This approach would maintain the explainability and precision of regexes while increasing their recall through the additional formulations. However, it was also mentioned that the crafting of regexes is laborious, time-consuming and error-prone, which makes this another aspect which can benefit from automation.

In this chapter, we propose an architecture of a regex detection tool that combines the previously introduced idea of word-embedding guided discovery and automated regex generation and eventually leads to a list of regular expressions which can be used for the detection of age discrimination.

Additionally, we incorporate concepts from active learning into the discovery process to reduce the amount of annotation, as stated in our original research goal.

We start this chapter with a background in automatic regex generation, followed by a description of our architecture in Section 5.2. Section 5.3 describes the evaluation procedure of our proposed architecture, followed by the results. We conclude with a discussion, including a list of possible improvements to the architecture.

5.1 Background

Regular Expressions are an efficient tool which is often used for Information Extraction. There exists numerous approaches to automatically generate these expressions (e.g., [30], [31], [32], [33], [34]). Automation of the regex creation process allows users with little or no knowledge of the regex syntax to generate regular expressions for a certain domain [31]. Further, it also reduces the amount of error which often occurs during the manual crafting of regexes and minimizes the creation time [35]. The architecture of the regex generation system developed in this study is based on several studies, which will be shortly introduced in the remainder of this section. Further, the concepts of active learning and genetic programming will be introduced.

5.1.1 Cluster-Based Regular Expression Generation

Babbar et al. propose a three-step cluster-based approach to regex generation [31]. Their implementation requires a dataset which is fully annotated for a certain entity of interest (ξ) and an initial regex as an input. During the first step, *Regex Relaxation*, they “widen” an input regex given by the user. Usually, users provide too specific regexes for the given problem domain, and this step assures that enough examples of the desired entity are found. This relaxation of the regex is achieved in two steps. Firstly, the regex expressions are widened based on the hierarchy of the regex character set. E.g., the `\d` character which matches digits only is relaxed to `\w`, which matches every alphanumerical character. Secondly,

quantifiers are widened by replacing $\{x, y\}$ by $\{x-m, y+n\}$, ($\{m, n, x, y\} \in \mathbb{N}$). To give an example, $R_3 = \backslash d\{2, 3\}$ can be relaxed to $R_{\text{relaxed}} \backslash w\{1, 4\}$.

During the second step, all matches to the relaxed regex are extracted from the input dataset. For each of the matches, a variation of the relaxed regex is generated which still matches the sample. E.g., the string $m_j = 1 - ABC$ was extracted with the regex from the last example. Following down the hierarchy of operators, it also matches the expression $\backslash d-[A-Z]\{3\}$, which can be obtained by reversing the relaxation again. Each of the matches is processed in this way and eventually clustered based on the regexes which were generated for the sample. This results in several regex clusters where the samples in a cluster all match the same regex. The entities of interest are processed in the same way so that eventually two set of clusters are obtained. One set contains all clusters of samples which matched the relaxed regex, and one set of clusters contains clusters of annotated samples (the entity of interest). The first set of clusters will at this point still contain noise. Therefore, lastly, set intersection is used to filter out all undesired clusters. For this, [31] propose to use a process of Greedy disjunction which subsequently eliminates noisy clusters until a certain user defined F-score threshold is reached. The regular expressions grouping the remaining set of clusters form the solution set of generated regular expressions.

5.1.2 Genetic Programming

Bartoli et al. present a series of papers in which they developed and improved a system to generate regular expression based on genetic programming. As the name suggests, this approach is motivated by its biological counterpart of Darwin’s theory of evolution.

Genetic Algorithms

Genetic programming is a sub category of evolutionary algorithms (also called genetic algorithms), an approach to solve search and learning tasks [36]. The task for a genetic algorithm can be represented as the following function:

$$f : S \rightarrow \mathcal{R}$$

where f is called *Fitness function*, S represents a candidate structure and R is the reproductive rate of structures S . The goal of the genetic algorithm is to find a structure S which maximizes (or minimizes) this fitness function. Starting with a pool of candidate solution, also called a *population*, the genetic algorithm manipulates all structures utilizing specific genetic operators. In one iteration, also called one *generation*, each candidate solution is manipulated once. At the end of each generation, the fitness of each of the structures is evaluated and used to select a number of structures which will be included in the next population.

Representation of Candidate Structures

The structures are also called *candidate solutions* and can be considered as such. They are possible solutions for the problem at hand. Usually, they are represented as strings. In case of *genetic programming*, the candidates are, in fact, small programs itself and the goal is to find the program which solves a specific program best. In this form of genetic algorithms, the candidate solution is usually processed in a syntax which is optimized for the problem [36].

For this study, this means that each candidate is a regular expression, and it is processed as such when the fitness of a candidate is evaluated. I.e., each candidate solution will be matched against a corpus and the quality of the extraction can be measured using, e.g., the F-score as a fitness function.

In terms of representation, candidates in genetic programming are usually represented as recursive tree structures, where each subtree represents an argument and its parameters [36]. For regular expression, this usually means that the regular expression operators are branch nodes and the leaf node represent

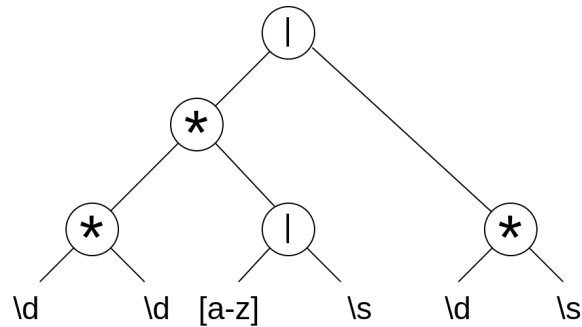


Figure 5.1: Tree representation of a regular expression. The asterisk (*) denotes a simple concatenation, the pipe operator (|) the or operator. This representation is equal to the regex $((\backslash d\backslash d|[a-z]\backslash s)|\backslash d\backslash s)$.

terminal nodes [32]. Figure 5.1 depicts such a tree representation. The benefit of this representation is that each sub-tree represents a valid solution to the problem, so the mutation of subtrees with the specific genetic operators will guarantee valid offsprings [36].

Operators

There are two main operators: *Mutation* and *Cross-over*. The mutation operation randomly mutates an element (or subtree) of a candidate solution by replacing it with another building block, much like genetic mutation changes a nucleotide in the DNA sequence. The Cross-over operator usually selects two candidate solutions and recombines them by swapping a subtree of one candidate with a subtree of another candidate, which results in two new, recombined solutions. The original trees are usually referred to as *parents* (ρ), the newly created trees are called *offspring*.

Tournaments

The process which selects the samples from the population which are to be modified is called *Tournaments*. Usually, seven random samples are picked from the not yet touched samples in a population, and the sample out of those 7 with the highest fitness is selected for the operation. If more than one sample is needed for the mutation (e.g., for Cross-over) the process is repeated as many times.

Parameters

Evolutionary algorithms have several parameters which can be varied based on the specific problem it aims to solve. The following table (Table 5.1) provides a short overview of them.

Term	Explanation	Default
Population size	the amount of candidate solutions which are manipulated each generation	<i>depends on problem space</i>
Mixing number	number of candidates used as parents during crossover	$\rho = 2$
Selection process	how a manipulation operation is selected for a candidate solution	<i>random or based on probability</i>
Crossover point	how to select the subtrees for the crossover operation	usually random if $\rho = 2$
Crossover rate	controls the amount of crossover operation per generation	<i>problem specific</i>
Mutation rate	controls the amount of mutation operation per generation	<i>problem specific</i>
Elitism	defines how the next generation is made up. Elitism assures that the highest scoring solutions will be included in the next generation	/
Culling	discarding of samples below a certain threshold	/

Table 5.1: Overview of Genetic Programming Parameters and Strategies [37, pp. 117]

Constraints and Evolutionary Algorithms

One weakness of evolutionary operators is that they are blind to constraints [38]. Therefore, in cases of constraint satisfaction problems (CSP), genetic algorithms do not appear to be the most obvious solution. However, Craenen et al.[38] showed that evolutionary algorithms can be used efficiently for solving constraint problems. They propose to add constraint handling to evolutionary algorithms to make them suitable for solving CSPs.

In essence, they present two ways in which constraints can be integrated into evolutionary algorithms, either through *indirect constraint handling* or through *direct constraint handling*. Indirect constraint handling means that the fitness function is adjusted in a way that it penalizes solutions which do not meet the constraints. Direct constraint handling requires the adaption of the evolution algorithm through, e.g., modifying the operators so that it is not possible to generate solutions which violate constraints. This technique is also known as the *preserving approach*. Further, it is also possible to include heuristics into the operators which allow to guide the evolution steps [38].

5.1.3 Using Genetic Programming for Regex Generation

As already mentioned, Bartoli et al. present an approach which utilizes genetic programming to generate regexes. Their approach aims to generate regular expressions without the need for an initial regex. Therefore, the user does not need to have knowledge of the regex syntax as opposed to the previously presented method by Babbar et al. [31]. In the following, we present the system design and the adaptations Bartoli et al. made to their system. They provide an implementation of the system presented in [34] at <https://github.com/MaLeLabTs/RegexGenerator>, which will be adapted for the regex discovery tool presented in the next section of this chapter.

Design

The system requires as an input several examples $X_i = \{\xi_1, \xi_2 \dots \xi_n\}$, which are tuples containing a sentence string t and a substring s of t which contains the desired extraction the solution regex should retrieve. Each candidate solution for the genetic programming is represented as a tree, where the leafs are

chosen from a set of terminal regex nodes and the branches are selected from a list of regular expression operators. The number of parameters of an operator determines the number of children for each branch node. These representations look as already depicted in Figure 5.1.

To evaluate the fitness of a candidate solution, the abstract tree representation is transformed into a string which represents the regex. This regex is then evaluated on the annotated test set, containing the tuples for all samples.

Evolutionary Search

Before the first generation, a population with randomly generated candidate solutions is created, using branches and terminals. During one generation, 10% of the candidates for the next generation are randomly generated. 80% of the candidates are manipulated using the crossover operator with $\rho = 2$ and 10% of the candidates are subject to random mutation. The samples for the crossover and mutation operation are selected in tournaments of size 7. After all samples in the current population have been manipulated, the samples with the highest fitness and the newly randomly generated samples are added to the population of the next generation. This process is repeated until either a perfect fitness function is achieved or the maximum number of generations is reached.

Fitness

Originally, the authors used a multi-objective fitness function, which aimed to minimize the Levenshtein distance and the length of the regular expression. Later, they improved their fitness function to a multi-objective fitness function that combines three metrics: *precision*, *accuracy* and *length* [34]. This multilayered fitness function ranks samples first by their precision. If the precision is equal, the accuracy decides about the ranking. And finally, if the accuracy is also equal, the length of the regular expression decides.

Separate and Conquer

A further improvement which was introduced at a later point is the *separate and conquer* strategy [34]. With this strategy, the genetic programming is encouraged to find the right amount of regular expressions. If, after the ranking at the end of the generation, one candidate solution achieves a precision score of 1, this solution and all samples xi that are correctly extracted by this solution are removed from the population and stored as an early solution. Once the evolution terminates, all early solution and the highest scoring solution from the last generation are concatenated with the regex pipe-operator `|`.

Pareto Fronts

The next changes of [34] over [32] improve the processing of the ranking. They introduce the notion of *Pareto domination*. s_1 is said to Pareto dominate s_2 , if s_1 is better in any one evaluation metric (precision, accuracy, length) and not worse in any other. All samples, which do not Pareto dominate each other but are dominated by the next higher frontier and dominate the next lower frontier, form one Pareto frontier. This improved ranking mechanism first groups all samples by their frontiers and then orders the samples within the frontiers, drastically reducing the amount of comparison of candidate solutions.

Further strategy improvements

In [34], Bartoli et al. propose an informed initial population. Based on the input strings and the desired extraction, the initial population is formed. Further, they also extract building blocks from these extraction strings if any common sequences are identified when generating candidate solutions. These building blocks are added to the regex alphabet and may appear in the randomly generated trees during a generation. Lastly, to encourage more structural diversity, they discard all except one tree if the string representation is the same.

5.1.4 Active Learning

In [39], Bartoli et al. introduce Active Learning to decrease the number of samples that are required to be annotated. The implementation of this was not provided publicly and goes beyond the scope of this work. We will only briefly explain the general concept of active learning, as the implementation described in the next section contains a basic active learning component

The key idea of active learning is to compensate the lack of annotated training data with strategic annotation of samples which are most beneficial for the algorithm to know the label of. There are several ways how to determine which samples need to be labeled, one of which is *Cluster-Based Active Learning*. Cluster based active learning, as presented in [40], makes use of the structure in the data by selecting data samples through considering, e.g., the density of the data at a certain point. This can help to select more representative data samples, compared to random selection of samples which may turn out to be outliers. For this cluster-based approach, the unlabeled data is clustered with an appropriate clustering technique (depending on the data). Once the data has been clustered, several samples from each cluster can be taken and added to a list of samples which need annotation. If the clustering was able to capture the data space successfully, this active learning method will assure that the small set of samples is representative for the dataset. One possible approach to pick samples from each cluster is to select all cluster centroids.

However, this also introduces a weakness to the approach. In case the clustering method fails to capture the space, the annotation set will not be representative and any further steps which rely on this selection will not be able to overcome this. Further, the choice of cluster numbers for dataset with unknown classes is always a hard one. The level of granularity can vary greatly and has an impact on the final performance of the model trained on the annotated dataset.

For this current study, we decided to follow the example of [39] and use active learning as an approach to address the lack of annotated data. But their implementation was beyond the scope of this study. We therefore settled for a basic implementation of active learning and separate it from the implementation by creating a module-based setup. The next section will introduce our proposed architecture.

5.2 Architecture

The regex discovery tool is a system designed to automatically generate regexes which should capture various formulations of so-called *key concepts*. A key concept could for example be “being a student” and the final regexes will match formulations like “I am a student”, “You are studying” etc.

This is achieved by combining a discovery, an annotation, and a generation module. From a top-level point of view, the discovery module helps to reduce the number of samples to a subset which can be annotated by using active learning techniques. This selection should contain as much different forms of discriminating samples as possible, as these form the basis for the regex generation. All forms of discrimination which are not discovered during this step will not reach the annotation and no regex will be formed matching them. The annotation module then keeps “the human in the loop” and assures that regexes are only generated for samples that are rated as discriminating by a human. Lastly, the generation module generates regexes which should match all discriminating samples and do not match the non-discriminating ones.

5.2.1 Input

Key concepts

The discovery module takes as an input a set of *key concepts*. For this study, we define a key concept to be one aspect of discrimination, e.g., addressing potential job candidates as students. The concepts need to be formulated as snippet of sentences. For the example given above, the key concept can be characterized by the following set of sentences:

```
(\^{}(?=.*tijdens)(?=.*studie).*)\\
(\^{}(?=.*je)(?=.*studeren).*)\\
(\^{}(?=.*studeren).*)\\
(\^{}(?=.*zoeken)(?=.*student).*)\\
(\^{}(?=.*voorkeur)(?=.*student).*)\\
(\^{}(?=.*medewerker)(?=.*zijn)(?=.*student).*)\\
(\^{}(?=.*vacature)(?=.*voor)(?=.*student).*)\\
(\^{}(?=.*tijdens)(?=.*je)(?=.*studie).*)\\
(\^{}(?=.*naast)(?=.*je)(?=.*studie).*)\\
(\^{}(?=.*je)(?=.*studie).*)\\
```

Listing 5.1: Sample formulation of key concepts

The quality of these formulations has direct impact on the discovery, the coverage the regexes will have in the end and the annotation effort needed. They should not be defined too narrow, e.g., by using really specific sentences. This will lead to a low syntactic variability in the samples that can be captured with the final regexes. On the other hand, too openly defined key concepts require a lot of annotation effort, as it increases the set of samples which will require annotation. The most extreme case would be to define key concepts in terms of only single keywords. This trade-off between annotation effort and coverage of the regexes needs to be considered when generating those formulations. More details about the effects of the formulations of the key concepts will be given in the next section when the discovery module is described.

Dataset

Additionally, a domain-specific dataset is needed. This data set should provide a good representation of the type of data the regexes will be used for. It needs to be composed of sentences, i.e., each sample in the dataset should be a sentence rather than a whole document.

If the regex generation tool is used for a time sensitive domain, it is important to choose a dataset with recent samples, since the regexes will be constructed based on phrases in the dataset. If, e.g., the domain for the regexes is youth language, a dataset from several years ago will not generate representative regexes for current youth language.

Language Model

Lastly, a language model is needed. This can either be a general pre-trained language model or a custom model trained for the specific domain.

5.2.2 Modules

As said, the regex generation tool has three modules: a discovery *module*, an *annotation module* and the *generation module*.

From a high-level point of view, the discovery module identifies a set of samples which are beneficial to be annotated. These samples will then be annotated by human annotators and afterwards used as input for the generation module which generates regexes. Depending on the annotation budget, the generated regexes can be verified by the annotators and refined with additional iterations of the generation process. Next to the set of regexes, this also generates a partially labeled data set, which can be beneficial for the development of further applications within the same domain.

The following sections will describe the different modules and submodules in more detail.

5.2.3 Discovery Module

The aim of the discovery module is to identify a set of samples from the input dataset which are beneficial to be labeled by a human annotator. For this, it takes the previously described formulations

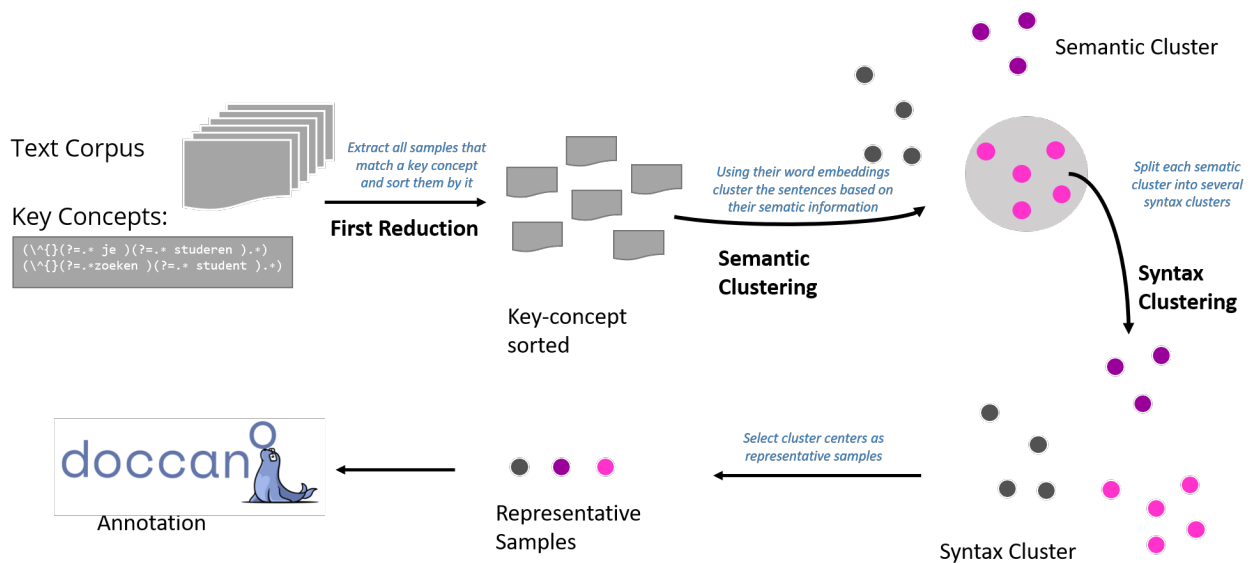


Figure 5.2: Overview of the Discovery Module Workflow

of key concepts as input to help to guide the discovery of relevant samples within a pool of sentences.

For the generation module, it is most beneficial to have a set of syntactically diverse samples for each key concept. Therefore, it is necessary to group all samples by their syntactic similarity and assure that sentences from all groups are selected. Syntactic similarity can be computed with the Ratcliff-Obershelp distance, a measure which computes the similarity of two strings based on common subsequences within those two strings. However, this measure requires the detection of common subsequences of two sentences, which is computationally heavy. Therefore, the discovery module groups the whole dataset into smaller groups with an additional grouping phase to reduce the amount of sentence which will be compared using the Ratcliff-Obershelp distance.

The discovery module can be described as a three-step approach, which includes a *first reduction*, *semantic clustering* using word embedding and *syntactic clustering* based on the Ratcliff-Obershelp distance.

Figure 5.2 gives a high-level overview of the workings of the discovery module, which we will describe in detail in the remainder of this section.

First Reduction

The first reduction phase is necessary to drastically shrink down a big dataset to a more manageable size to perform more computational heavy operations on them. The current implementation first filters out all sentences which contain the stem of one of the keywords and groups them by the keyword occurrences. Sentences which contain more than one keyword will occur in both sets. This reduced set can then be lemmatized. If the initial dataset is small enough or enough computational resources are available, the whole dataset can be lemmatized immediately. As a second step, the lemmatized sentences are further reduced to only those sentences which contain the lemmatized key concepts in any possible order. For this, the tokenized and lemmatized key concepts are reformulated into an order independent regex (see Listing 5.1).

To sum up, the first reduction module filters out all sentences which contain the same main stem as the key concepts and groups them by keywords. All sentences which do not fulfill this criterion are discarded at this point. This highlights why the formulation of the key concept is such a crucial step for the coverage of the regexes. If only a narrow set of key concept with several words per concept is

chosen, the resulting set will be more restricted as compared to using only keywords as key concepts. Further, wider key concept formulation also increase the number of context the keyword can be used in. Since eventually the number of different formulations per context will indicate the minimal annotation number, the annotation effort also depends on the formulation of the key concepts.

Semantic Clustering

The first reduction results in a grouping of samples based on the occurrence of the different formulations of the key concepts, with one group corresponding to all samples that match the respective key concept formulation. Depending on the size of the input data set and the frequency of the key concepts in the data, these groups may, however, still be too big to efficiently compute a distance matrix based on the syntactic similarity of the sentences using the Ratcliff-Obershelp distance. The goal of the semantic clustering is therefore to further split these clusters by their semantic meaning. This is done by using embedding representations of the reduced dataset (or full dataset if the set is small enough/ enough resources are available).

The regex discovery tool allows defining a cluster-size threshold. All key concept groups which have more samples than this threshold will be semantically clustered. All key concept groups with fewer samples will be passed on to the syntax cluster submodule without further semantic grouping. The semantic clustering is done based on the vector representation of the sentences generated by the input language model. The representations are reduced in dimensionality using UMAP and clustered. Currently, Agglomerative clustering is used for clustering, but this may be done by any other clustering approach suitable for the data at hand.

The semantic clustering submodule eventually produces a set of semantic clusters which form groups within each of the key concepts.

Syntax Clustering

The aim of the syntax clusters is to group the data into small groups, where each group represents one possible lexical formulation of a certain key concept. To generate regexes with wide coverage, we need to obtain a collection of sample sentences which contains a diverse set of formulations of the key concept. To do so, the syntactic distance between all sentences within a semantic cluster is computed. This is done using the Ratcliff-Obershelp distance, which is based on the occurrence of common subsequences in two sentences. Since regexes are order-dependent, sentences which have the same order of words can be easier included in one regex than sentences where the word order of certain keywords differs. Based on the computed Ratcliff-Obershelp distance, smaller syntax clusters are formed. In the ideal case, each cluster represents one specific sequence of words which can be captured with one regular expression and only contains sentences which either represent the key concept or do not represent the key concept. In other words, this clustering aims to get somewhat clean clusters in terms of the key concept label which are syntactically similar. To give an example, in the ideal case we try to get all formulations which express that someone is a student. E.g. “you are a student” and “you are studying” are lexically more similar than “are you finishing your studies?”, since the first two sentences have more words in a row in common.

Sample Selection

As a last step, these clusters are used as the foundation for the selection of the samples which will be passed on to the annotation module. The selection, which is done here, depends on the given annotation budget. For the current implementation, we decided to extract one sentence per cluster for the annotation. To obtain the most representative sentence from each cluster, we extract the sentence which forms the cluster center, in case the agglomerative clustering approach converged. But for cases with a bigger annotation budget, the implementation could also be extended following the pseudocode:

```
sort cluster by cluster member amount (decreasing size)
```

```

for each cluster:
    select center
while not annotation budget reached:
    for each cluster:
        select sample that maximizes distance to selected samples(s)
        from same cluster

```

This follows the idea of selecting samples in active learning. Once the set of samples for annotation has been generated, they are written into a .txt file.

5.2.4 Annotation

For the annotation, we once again picked Doccanno as our annotation software (already introduced in Chapter 3). The interface is easy to use and allows annotators to annotate the provided samples. This time, we choose the NER (Named-entity-recognition) annotation mode in which annotators can highlight those portions of the text that contain the ageist phrase, which will later on help to generate the regexes.

Regarding the requirements we outlined previously, this step assures that all generated regexes are based on sentences which were annotated by humans. Thus, it is possible to take a certain legal framework into consideration which the regexes should comply with.

The annotators have the choice between two different labels: *discrimination* and *ignore*. In case the sentence does not contain any age discrimination, the sentence remains blank. They are asked to annotate parts of the sentences they identify to be age discriminating with the *discrimination* label.

The *ignore* label allows excluding sentences from further processing. Reasons for this could be that the annotator is not sure about the label itself, or the sentence should be excluded for any reason. This assures that sentences for which the annotator cannot find a label are not automatically considered as non-discriminating sentences, but they are removed from the set of samples. The sentences marked with “unsure” can either be revisited at a later point, kept out of the dataset completely, or possibly be replaced with another sample from the same cluster. For the current implementation, we choose to remove them from the set, as the aim of the study is to prove the concept of this architecture.

Once the annotator(s) finished, the annotated dataset can be downloaded in JSON format for further processing.

Preparing for Regex Generation

The annotated dataset will need to be processed so that the input format matches the required input of the Genetic Search module. Firstly, the samples with *unsure* are filtered out. Afterwards, the substrings which were identified as being discriminating are extracted.

5.2.5 Genetic Search Module

The genetic search module follows closely the implementation of [34], which provided the source code for their regex generation implementation at <https://github.com/MaLeLabTs/RegexGenerator>. This implementation did, however, not fully meet our needs, as it was mainly developed for information extraction of entities, such as IP addresses, URLs, phone numbers and the likes, all of which are more character based. The domain of regex generation for matching sentences allowed us to consider the smallest mutable entities in the candidate regular expressions to be token/words rather than characters. We, therefore, decided to re-implement a simplified version of the provided implementation, which is adapted to the properties of sentence matching. Out of convenience and to be able to enhance the implementation through NLP frameworks such as NLTK and spaCy at a later point, we decided to implement the Generation module in python.

Population Setup

Following the original implementation, we generate a non-random initial population by transforming the substrings which should be extracted into abstract tree representations. The original implementation by Bartoli et al. generates four different tree structures per positive sample (sample which has a desired substring), which could be a possible improvement for the next version of the Regex Generation Module.

Alphabet

Based on the tokens generated for the initial population, we create an alphabet, which contains all possible leaf nodes. Additional to the tokens from the desired substrings, we also allow the user to add custom regex terminal notes such as `\w`, `\d`, `[A-Za-z]+`. Further, all tokens from the initial population are stemmed, and their possible suffixes are added as optional characters to the stem. This means that e.g., the words *student*, *studenten* and *studeer* will be expressed as the additional terminal node `stude(nt|er|nten)`, as there are no operators which allow the modification of terminal nodes yet. This alphabet will be used by the mutation operator.

Operations

We choose to implement 4 operations. The traditional operators *Crossover* and *Mutation* are implemented as described previously. Based on the success of adjusted operators in Genetic Programming [38], we choose to adapt the two traditional operators. In a true CSP case, the adopted operators would replace the traditional operators, as those will not always produce solutions which meet the constraints. In our case, however, we use the adapted operators to introduce some prior knowledge to the generation to shape the evolution.

The two new operators are inspired by the traditional operators, just more restricted in their choice of transformation. For what we call the *Or-Crossover* operator, we extract two subtrees from two parent nodes, as it is done for the crossover mutation. But instead of swapping them and adding them to the other parent, we concatenate the two subtrees with an *or* operator and add this combined or-tree to both parents.

Figure 5.3 depicts this process. The second new operator was inspired by the mutation function, but instead of mutating one element of the selected tree with a random node, the *wildcard mutation* operation only selects leaf nodes for the mutation and replaces them with a wildcard operator `[A-Za-z]+` which matches any word.

Strategies

In terms of evolution strategies, we implemented the separate and conquer technique [34]. However, during initial testing, we discovered that the early termination criteria of $Prec = 1$ lead to nearly exact matches of the initial population strings. We therefore decided to adjust the Precision function as described in the next paragraph.

Fitness

Due to performance issues with our implementation, we choose to implement a less granular implementation of the fitness function. Instead of basing the fitness measures of precision and accuracy on the extracted tokens and the desired extraction, we measured the precision and recall regarding positive and negative samples. That means, we considered a positive sentence (a sentence which contained a desired extraction) to be TP if any candidate solution extracted some substring from this sentence. Following this logic, FP describes a negative sentence for which a candidate solution extracted something, FN a positive sample for which no candidate solution extracted anything and TN a negative sample for which nothing was extracted. To address the overfitting problem described in the last paragraph, we added

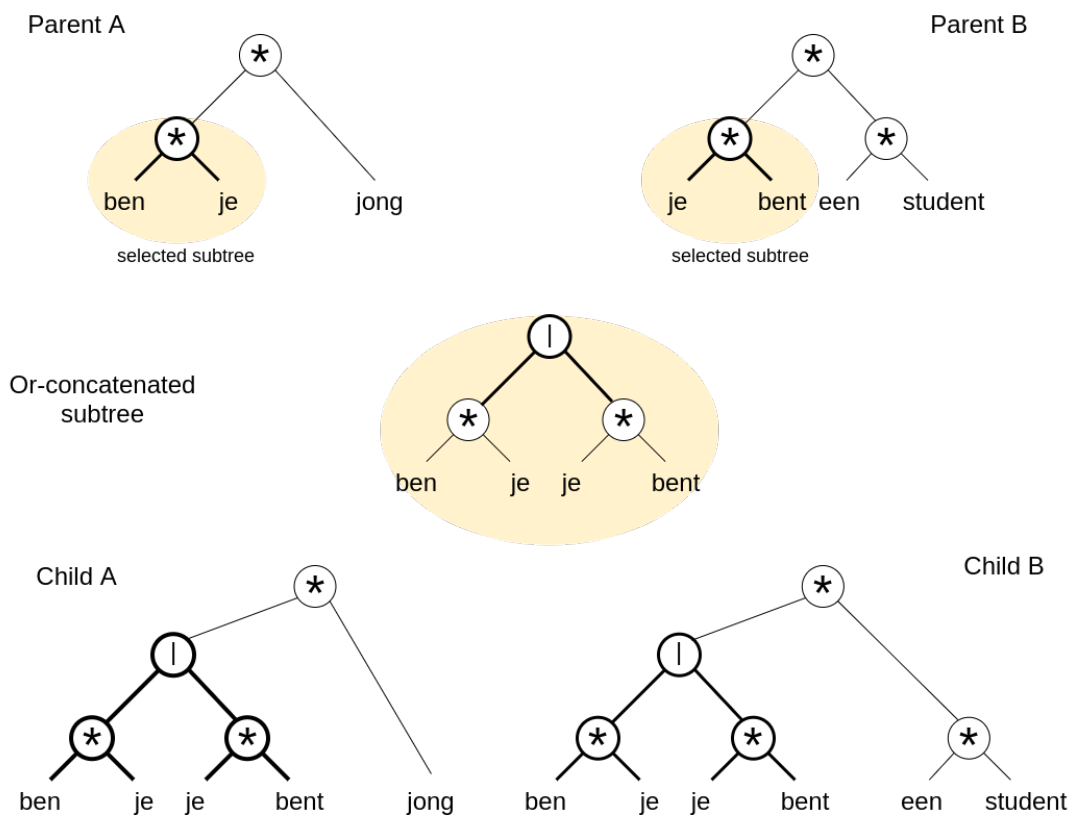


Figure 5.3: The or-crossover process

a penalty term to the precision calculation, which penalizes low TP count. We define the precision as follows:

$$\text{prec} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5.1)$$

$$\text{prec_adjusted} = \text{prec} * \frac{1}{1 + e^{-\text{TP} * \lambda}} \quad (5.2)$$

λ is a user-specific error term that controls the number of needed TP for early solutions. It should be chosen based on the amount of positive samples in the dataset, with lower values for λ for more required TPs. This adjusted required a change in the condition of early solutions, since $\text{prec_adjusted} = \text{prec} * \frac{1}{1 + e^{-\text{TP} * \lambda}}$ is asymptotic to $x = 1$. We lowered the threshold to be $\text{Prec} = 0.99$. This means, however, that it could happen that some early solutions may contain False Positives. But following the intuition of a warning system implementation for regex detection, we decided to focus on recall rather than precision here, as the introduced probability for False Positives is low.

Parameters

The genetic search module takes several parameters as an input, which need to be provided in a config file. The following parameters can be specified:

- Crossover rate
- Mutation rate
- Regex rate
- wildcard rate
- or rate
- n_gen
- n_pop
- n_stop
- λ

Here, the different rates determine the likelihood with which one operation is selected for a sample. n_gen denotes the number of generation, n_pop the population size and n_stop the maximum iterations without change before the search terminates.

Output

The final output of the regex generation tool is a list of regexes generated through for each of the individually issued genetic searches. The number of regexes which are discovered per genetic search highly depends on the input data and the selected parameters. Further, the genetic operations are applied at random and a genetic search is not guaranteed to find the optimal solution.

5.3 Experiments

The main goal of the evaluation of the regex generation tool is to research if it can provide valid regexes, and evaluate how the regexes perform regarding precision and recall. We used the previously described implementation of the regex generation system with the following configurations.

5.3.1 Data

As input data for the regex discovery tool, we used the *2014 job description dataset* and the Language Model described in the previous section, which was trained on this particular dataset.

5.3.2 Keywords and key concepts

We decided to use the regexes from the Dutch Baseline as guidelines for the generation of the key phrases, in particular the indirect discrimination regexes list. We did this to have a baseline to compare our results with. Further, it would also make it possible to compare human made regexes to machine-generated regexes, which could give further insight in potential shortcomings of machine-generated regexes. A list of the key concepts used can be found in Table 5.3.2

5.3.3 Discovery Module

We used all three steps of the discovery module, first reduction, semantic clustering and syntactic clustering.

5.3.4 Annotation

To fit the goal of minimal annotation time, we set the annotation size for each semantic cluster to be 1, which were annotated by the researcher of this study.

GP parameters

We were unable to process all samples at once due to computational limitations. Therefore, it was necessary to split the annotated sentences into several subgroups. I.e. we were unable to use the whole annotated dataset as input to the regex generation module and it was necessary to split it into several sets which would be processed independently. The challenge here was to assure that similar ageist phrases would end up in one cluster to increase the likelihood that a solution would be found. We solved this by clustering all samples with a positive age discrimination label based on the identified age discriminating snippet in the sentences. This was done using once again Agglomerative clustering with the Ratcliff-Obershelp distance, resulting in 11 clusters. To each cluster, we added all negative samples (samples without age discriminating snippets), even though which matched other key concepts, to assure that no regexes would be generated that matches non-discriminating samples.

We processed all the clusters with the parameters listed in Table 5.3.4.

5.4 Evaluation

We evaluated the performance of the regex detection tool by looking at the intermediate results from each module with the settings described above and by comparing the performance of the generated regular expressions with the indirect flexible regex list from the Dutch line. For this, we applied both sets of regexes (the generated ones and the baseline regexes) on the set of annotated samples we used as input for the generation module and the *annotated dataset*, which was introduced in Chapter 3. We assessed the performance by using Precision, Recall and F-Score.

5.5 Results

We split the presentation of our results in two parts, the intermediate results which include the results from the different modules within the regex generation tool and the results from the evaluation of the generated regexes.

Keyword	Key Concept
bijverdienen	bijverdienen naast je studie
bijbaan	bijbaan leesrooster bijbaan naast studie bijbaan naast school
starter	ben je starter ben u een starter zoeken starter op zoek naar starter vacature for starter ben starter
stap	eerste stap carrière tweede stap carrière
ervaring	max jaar ervaring maximaal jaar ervaring
start	starten loopbaan een start bieden
leesrooster	leesrooster passen
baan	eerste baan
stud	tijdens studie je studeert studeer zoeken student voorkeur student medewerker zijn student vacature voor student tijdens je studie naast je studie je studie
opleiding	volgt opleiding bezig opleiding
jong passen	jong passen
jong uitmaken	jong uitmaken
jong versterken	jong vesterken

Table 5.2: List of input key concepts by keyword

Parameter	Value
crossover rate	0.1
mutation rate	0.15
regex rate	0.1
wildcard rate	0.5
or rate	0.1
n_gen	5000.0
n_pop	500.0
n_stop	500.0
λ	1.0

Table 5.3: Parameters for the Genetic Search Module

Keyword	Semantic	Syntax
bijbaan	2	10
lesrooster	6	47
opleiding	28	419
stap	13	97
start	6	47
stud	34	335

Table 5.4: Overview of the number of semantic and syntactic cluster per input keyword

5.5.1 Intermediate Results Regex Discovery Module

Sample Discovery

With the given input data and key concepts, the discovery module discovered a total of 38242 samples which contained any of the key concepts. These samples were clustered into a total of 35 semantic clusters and 955 syntax clusters, meaning that 955 sentences were selected as representative samples that would need to be annotated. For the keywords *stud* and *opleiding* the syntax cluster produced too big clusters for further processing. As an intervention, we used the K-Medoids Clustering approach to split the big cluster into several small ones. With this technique, we would not need to discard any samples to make the computation expedient. The only potential drawback we expected was that we would obtain several quite similar samples for the keywords *stud* and *opleiding*.

Cluster number	Sample Count
0	12466
1	361

Table 5.5: First Cluster results for *opleiding*

Cluster number	Sample Count
0	120
1	605
2	13570

Table 5.6: First Cluster results for *stud*

Table 5.5 shows the clustering for *opleiding* before intervention. Through the K-medoids clustering, we split cluster 0 into 27 smaller clusters. Table 5.6 shows the clustering for *stud*, for which cluster 2 eventually was split into 34 new clusters.

Annotation

We annotated 955 samples. In total, 425 samples were annotated as negative (did not contain an ageist phrase), for 433 samples an ageist phrase was identified within the sample. For 97 no label could be assigned and they were discarded. Reasons for discarding were either noisy samples or that no clear label could be assigned based on the sentence.

5.5.2 Genetic Search

After clustering the positive annotated samples, which was needed due to computational limitations explained above, we obtained 11 input sets for the genetic search module. The genetic search model generated 27 regexes in total. Table 5.7 gives an overview of the number of obtained regexes for each of the independent genetic search runs. The generated regexes can be found in Appendix B.

5.5.3 Evaluation of the Generated Regexes

Correctness of the generated regexes

The output generated by the regex discovery tool could be compiled to regexes through the regular expression package `re` for python, which means that only valid regular expressions were produced.

Cluster Number	Number of found regexes
0	2
1	13
2	1
3	1
4	1
5	1
6	1
7	2
8	1
9	1
10	1
11	2

Table 5.7: Overview of obtained regexes per GP run

	True	False
True	135	275
False	0	448

Table 5.8: Confusion matrix for the generated regexes on the genetic search input samples

	True	False
True	75	335
False	10	438

Table 5.9: Confusion matrix for the baseline regexes on the genetic search input samples

Evaluation: genetic search input sample set (training set)

Table 5.8 shows the confusion matrix for the generated regular expressions on the samples which were used as input for the genetic search module. In comparison, Table 5.9 shows the confusion matrix for the flexible indirect regexes from the baseline on the same sample set.

Additionally, Table 5.12 compares the precision, recall and F-score values for both sets of regexes. The generated regexes reach a perfect precision on the input dataset, the baseline regexes only reach a precision of 0.88 on the same sample set. Contrary to the high-precision values, both regex sets struggle with recall. Here, the generated regexes only reach a precision of 0.329 and the baseline regexes have an even lower recall of 0.183.

Evaluation: *annotated dataset* (test set)

Table 5.10 and Table 5.11 show the confusion matrix on the annotated dataset. Table 5.13 compares precision, recall and F-score values for both sets of regexes. On the *annotated dataset*, the baseline regexes outperform the generated expressions. Both regex lists reach a high or rather high recall (0.985 for the baseline regexes and 0.827 for the generated regexes). But in terms of recall, the baseline scores much higher than the generated regexes, with a recall score of 0.805 for the baseline and a score of only 0.177 for the generated regexes.

5.6 Discussion

Below we will discuss our findings considering the quality of the regexes and afterwards address the shortcomings we highlighted by possible improvement strategies.

5.6.1 Quality of the generated regexes

The most prominent observation of the results is the low recall of the generated regexes. Both on the genetic search input sample set and the annotated dataset, the generated regexes can only detect a

	True	False
True	205	956
False	43	1606

Table 5.10: Confusion matrix for the generated regexes on the *annotated dataset*

	True	False
True	935	226
False	13	1636

Table 5.11: Confusion matrix for the baseline regexes on the *annotated dataset*

	Generated	Baseline
Precision	1	0.88
Recall	0.329	0.183
F1-Score	0.248	0.152

Table 5.12: Precision, Recall and F-Score for genetic search input samples

	Generated	Baseline
Precision	0.827	0.986
Recall	0.177	0.805
F1-Score	0.145	0.443

Table 5.13: Precision, Recall and F-Score for the *annotated dataset*

fraction of the discriminating samples sufficiently. When looking for an explanation for this low recall, there are two points to focus on, the discovery module and the generation module.

One reason for the low recall on the annotated dataset could be that the discovery module did not produce samples which represented the whole corpus well. However, the recall on the genetic search sample set is also low, which means that the regexes do not even capture all input sentences. Since the recall on the genetic input dataset is slightly higher than on the annotated dataset, it seems likely that selected samples do not fully represent the whole corpus. But we also choose to only annotate the absolute minimum of samples for the annotation set. Further research is needed to see if the recall on the training (genetic search input) and validation set (annotated dataset) can be moved closer together. Thus, the discovery module may have contributed partially to the low recall, but is not fully responsible for it. It seems more likely that the genetic search algorithm was unable to generate regexes which fully covers the input data, i.e., the approach failed to generalize. This hypothesis is also supported by the observation that no early solutions were found and the general number of regular expression generated is low. A possible explanation could be that the genetic search module did not converge during runtime and therefore did not produce regexes with coverage over the whole input set.

The generated regexes are, however, able to score perfect and somewhat high precision on the training and test set respectively. The lower precision on the test set is not too surprising, however, given that the human crafted baseline regexes score higher on the test set, there is still room for improvement.

We therefore have to conclude that we did not reach our goal of constructing an architecture which is capable of producing high recall and precision regular expressions. Especially the low recall requires further improvements on the architecture. But since the regex detection tool was able to produce valid regular expressions with a somewhat high precision, we are optimistic that with further improvements it might be possible to utilize machine learning to improve the generation of regexes to detect age discrimination. Below, we will elaborate further on these possible improvements.

5.6.2 Possible Improvements

The current results show that the regex detection tool is not yet fit for efficiently generating regexes which provide good coverage of the input data. In the following, we will describe several strategies which could be implemented to improve the current implementation. We will start with improvements for the discovery module and continue with the genetic search module.

Improvement Suggestions for the Discovery Module

Section 5.6.3 highlighted the dependency of the selected samples for annotation and the quality of the regular expression. Future research could be done on improving this selection.

This could be achieved either by changing the clustering approach or the selection strategy. I.e., depend-

ing on the annotation budget, more samples could be selected, as already proposed in Section 5.2.3.

Another possible improvement would be to open up the first reduction phase and replace some words from the key concepts with synonyms to increase the coverage. Further, it might also be possible to replace the current implementation and make use of probabilistic labeling, an approach which enhances the labeling of big unlabeled dataset.

Lastly, it would also be worth exploring if those probabilistic labeling approaches are suitable to generate noisy labels for the active learning, which could be used as input for the genetic search. We reason that an improved discovery model will increase the coverage of different manifestations of age discrimination and thus the overall coverage of the generated regexes, if the genetic search component will also be improved accordingly.

Improvement Suggestions for the Genetic Search Module

The first step in improving the genetic search module should be the efficiency of the implementation, which would facilitate the evaluation of other improvements.

Once the efficiency has been improved, the fitness function could be adjusted. Currently, it does not consider which tokens are extracted from a sentence, but only if anything is extracted at all. This could be changed to a token-based evaluation. We believe that this would improve the quality of the regexes, given that the annotation quality is high. Further, more heuristic-based operators could be added. These operators could even utilise syntactic properties of the input sentence, i.e. the part of speech of a sentence.

Also, synonyms could be detected which would allow grouping words such as *studie*, *opleiding* and create combined terminal nodes of the style (`studie|opleiding`). We believe that adding more prior knowledge could potentially increase the performance of the regex detection tool. However, it needs to be considered in which way this knowledge is added to the system. Here, it will be necessary to differentiate between domain knowledge and purely syntactic knowledge. Knowledge based on grammar could be directly implemented into the operations. With domain knowledge, e.g., specific properties of the discrimination form, would need another way to be introduced to the system so that it can be easily adapted to different forms of discrimination. In the ideal case, it should be possible that the domain knowledge is fed into the systems in a way that does not require any programming knowledge. This would allow domain experts to directly interact with the system without the need of technical support.

General Improvements

Looking at the system as a whole, more testing needs to be done to improve the communication between both modules. For this study, we focused on one type of discrimination, but it might be more beneficial to consider one keyword isolated rather than trying to discover samples for several keywords.

Different domains

Lastly, it would be interesting to apply the Regex Generation Tool to other domains. This could be other forms of discrimination, but also the generation of regexes for other domains, which detection can be broken down to a list of regexes based on some input concepts (key concepts). Further, different language inputs can also be thought of.

5.6.3 Limitations

The results clearly highlight that this approach is not yet fit to be used as the pure source of regexes for discrimination detection tasks. It should therefore be considered more of a proposal architecture rather than a complete system. Further, we want to address certain limitations of this approach, which will remain even with increasing quality of the output.

Firstly, there is a clear dependency on the quality of the input data. We already highlighted how the selection of the key concepts influences the selection of the samples for annotation. On the one side, this is what makes this architecture so general, on the other side this puts a lot of emphasis on the user input.

Secondly, annotating the data controls which samples will be used for the regular expressions. The amount of data to annotate is drastically reduced for the proposed system, which puts a lot of stress on the quality of the annotation. As with the first limitation, this decision was made to allow this system to be used in a general way and additionally to give the decision power to a human. This power, however, also comes with the responsibility to create high-quality annotation which will have a substantial impact on the final set of regexes. Thirdly, any set of regexes which will be generated may suffer to some extent from the shortcomings we highlighted in Chapter 3. Even with an optimal selection of samples for the annotation and a well performing genetic search module, there can always be new formulations which are not considered by the regexes. If the system is improved and the input and annotations are of good quality, we hope to minimize the effect of the shortcomings, but there are with near absolute certainty always samples which cannot be identified or other samples which may be detected mistakenly.

However, this approach as opposed to pure machine learning approaches gives the end user more control and allows for correction at nearly any point in the system. The human is constantly kept in the loop, which places a lot of responsibility on them, but also allows to influence the system output directly in a comprehensible way.

5.7 Conclusion

In this chapter we addressed our third research objective. Taking the findings of our two previous sub-studies in Chapter 3 and Chapter 4 into account, we presented a possible architecture that allows the automatic generation of regexes to detect age discrimination. The evaluation of our architecture revealed that it is not yet fit for generating regexes with enough coverage which can be used to sufficiently detect regexes. But we provided a list of improvements which we hope addresses the issues of the low recall of the generated regexes and might yield in a system which is capable of generating high-quality regular expressions for discrimination detection and similar domains.

Chapter 6

Conclusion

In this study, we explored the possibilities of using more advanced AI techniques to automatically detect age discrimination in Dutch job advertisements, with the additional limitations of high explainability and low human annotation. Due to the limited research, we decided to approach the problem through two exploratory studies which would (1) show the strength and the weaknesses of the current regex-based Dutch baseline and (2) explore in which way objective age discrimination is captured by the latent space spanned by ALBERT word embeddings.

We found that regexes reach a rather high precision and provide an explainable solution to age discrimination detection. However, crafting regexes is time intense and the first study revealed that the current Dutch baseline lacks coverage over all the possible formulations in which objective age discrimination occurs.

The findings of the second study suggest that age discrimination is not captured in the latent space spanned by ALBERT word embeddings which were trained on a domain-specific job description corpus. We did, however, observe evidence that the context of age discrimination suggesting keywords is captured and if this context is inherently discriminating, clusters according to discrimination labels were found.

These findings informed the architecture of an age discrimination detection approach, which utilizes the explainability and precision of the regexes but automates the regex-generation approach through AI techniques.

The final approach was not able to reach generalization, but all generated regexes were valid and performed high precision on the test set. We discussed the shortcomings of our approach and listed several points of improvements for further research.

To conclude, this thesis proposes an exploration-informed architecture to automatic age discrimination detection, which is not yet fit for real-life usage but provides a first prototype to a concerning problem which is not yet addressed sufficiently by research.

Appendix A

Annotation Guide



Annotation Guide

Hello,

this is your annotation guide for labeling job descriptions with regards to age discrimination.

First of all, thank you for helping me out. The annotations I am collecting here are important for my master thesis results, I truly appreciate your effort!

To get you ready for the annotation, I have written an annotation guide explaining what you will need to do. I will start with a short introduction to the annotation task, give you an introduction to the annotation tool and then go into more detail about the annotation itself. In the end, you'll find a short paragraph on what my thesis is about and why I need the data, in case you are interested

The task

You will be labeling sentence from job descriptions. These sentences were automatically extracted and some may look a bit odd or cut off, but you should be able to understand what the sentence(s) are saying. You will need to label each sentences from a set of labels indicating if the sentence entails any age discrimination. Except for specific cases (which are stated in the annotation section), you will need to select one label per sentence.

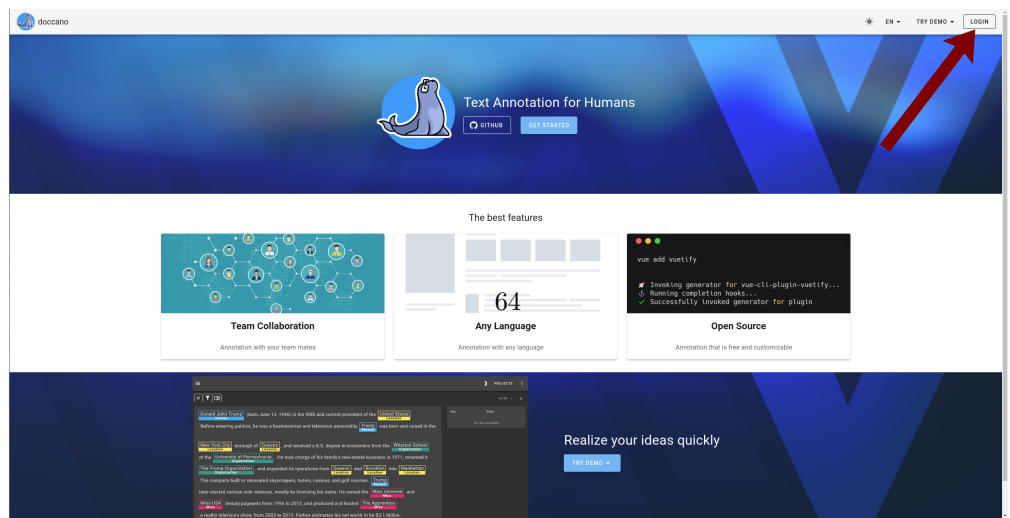
The Tool

The annotation will be done in a tool called "Doccano". You can access it here:

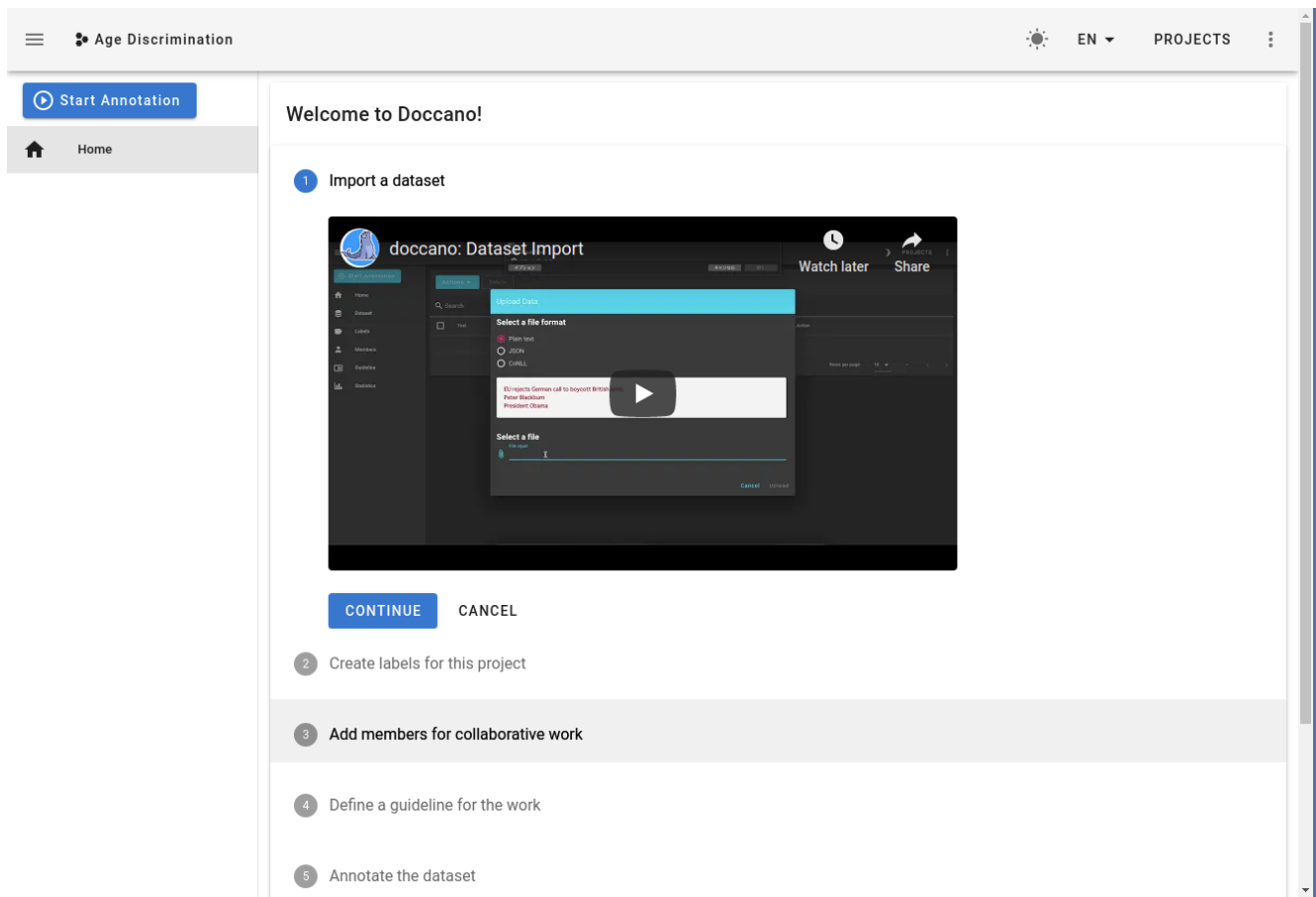
```
doccano - doccano
doccano is an open source annotation tools for machine learning practitioner.
https://agedislabels.azurewebsites.net/
```

The interface

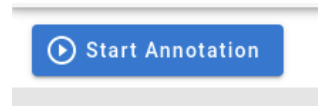
Please select [Login](#) (its a bit hidden in the top right corner)



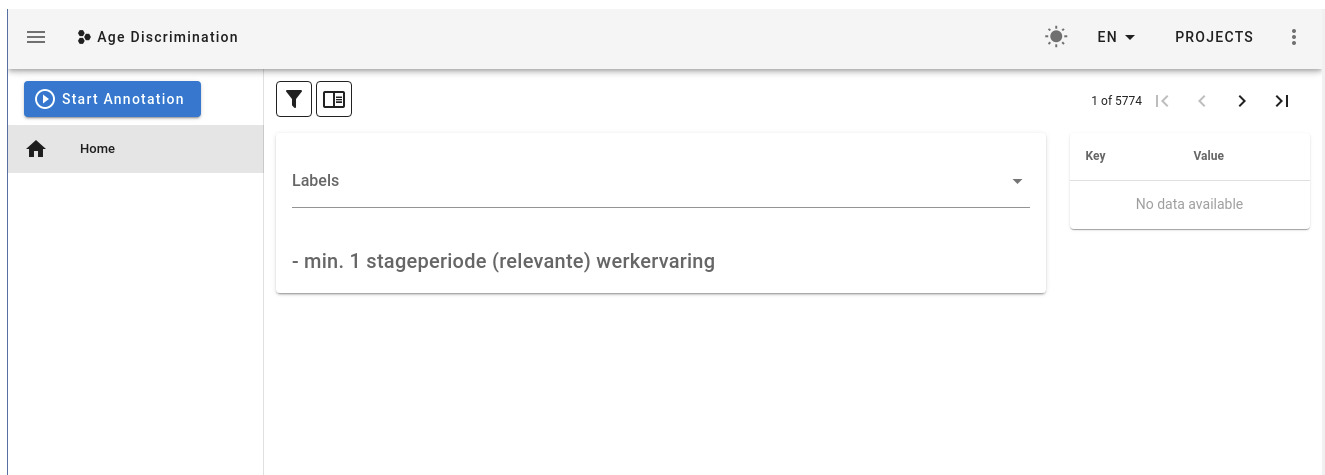
After logging in, you will be asked to select a project. There is only one, so just select it. You will land on a page that looks something like this:



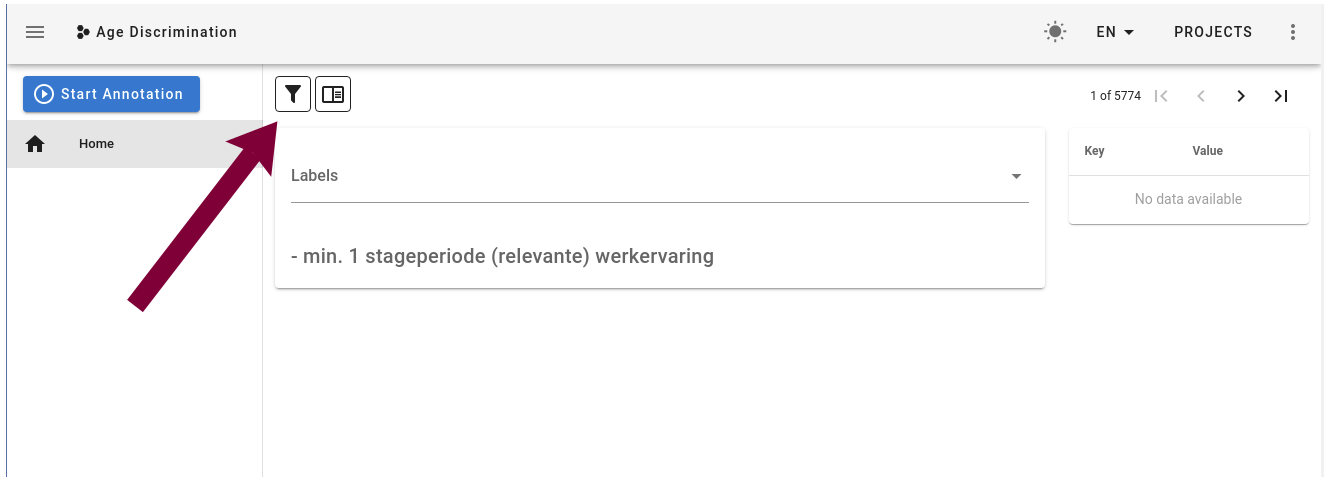
Please go to the **Start Annotation** button on the top left hand side.



The labeling page



Before you start labeling, please filter the data to not yet annotated data. You do this by clicking the filter icon and select **undone**. This way you won't get any already annotated data.



In the middle of the screen you see the current sentence. Use the dropdown menu **Labels** to select the respective label(s) for the sentence. You can also just use keyboard shortcuts. Unfortunately, they are not displayed on the site itself, so I attached a file with an overview image you can have open next to the annotation window. The shortcuts are intuitive, so don't worry.

After you have selected the label for the sentence you can navigate to the next sentence either with your keyboard arrow keys or with the arrows on the top right site of the screen.

That's all to it.

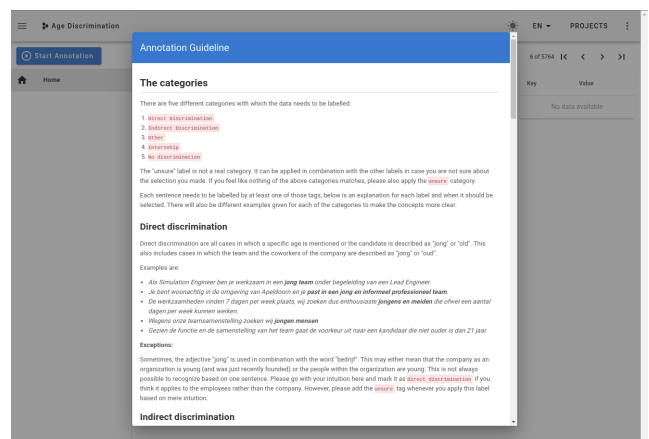
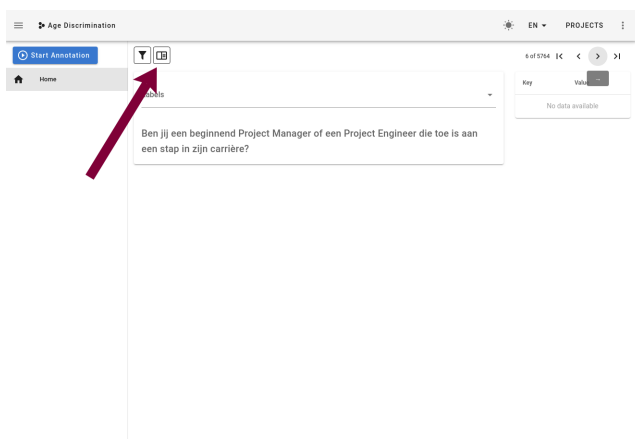
The number you see on top is the number of all samples in the dataset. But don't worry. The whole team is working on it, you will only get samples which have not been rated yet. I am thankful for any sentence you label for me 😊.

Continuing after leaving the tool

You can stop at any given time and close the window/log out. Your progress is saved the moment you select a label. If you want to continue afterwards, you'll jump right to the next unrated sample.

The annotation guide

Below you find an overview of the categories and instructions on how to label the samples. In case you want to refer back to this guide you can do so from the labelling platform itself:



Clicking the right button on the top of the screen will show you exactly the same guidelines you see below.

⚠ Sometimes, if you click too fast you might get a 403 error code pop-up. Just refresh, and all is good to go again

The categories

There are five different categories with which the data needs to be labelled:

1. **Direct Discrimination**
2. **Indirect Discrimination**
3. **Other**
4. **Internship**
5. **No discrimination**

unsure

The "unsure" label is not a real category. It can be applied in combination with the other labels in case you are not sure about the selection you made. If you feel like nothing of the above categories matches, please also apply the **unsure** category.

Each sentence needs to be labelled by at least one of those tags, below is an explanation for each label and when it should be selected. There will also be different examples given for each of the categories to make the concepts more clear.

Direct discrimination

Direct discriminations are all cases in which a **specific age** is mentioned or the candidate is described as "**jong**" or "**oud**".

Examples are:

- *je bent tussen de 16 en 27 jaar oud*
- *De werkzaamheden vinden 7 dagen per week plaats, wij zoeken dus enthousiaste **jongens en meiden** die ofwel een aantal dagen per week kunnen werken.*
- *Voor deze functie zoeken wij een jonge enthousiaste medewerker met:*
- *Wegens onze teamsamenstelling zoeken wij **jongen mensen***

By law, they are certain age groups which can be discriminated by, e.g. younger than 18 years (often due to child protection laws). Please still mark all samples of minimum age requirements as discriminating, no matter the age group.

Special Case: "Jong team"

There are a lot of samples, where the team is described. These sentences are a bit ambiguous. If they are phrased in a way like:

- *Gezien de functie en de samenstelling van het team gaat de voorkeur uit naar een kandidaat die niet ouder is dan 21 jaar.*
- *Bij ons maak je deel uit van een jong interactief team met uiteenlopende disciplines - dat team wordt alleen maar sterker met jouw bijdrage.*
- *Je bent woonachtig in de omgeving van Apeldoorn en je **past in een jong en informeel professioneel team.***

it is direct discrimination, as the applicant is described through them. If however, they describe the team/company they have with the word "jong" as in this sample:

- *Als Simulation Engineer ben je werkzaam in een **jong team** onder begeleiding van een Lead Engineer.*

please use the label **Other**. This also holds for samples such as:

- *De opdrachtgever is een jonge onderneming*
- *Je toekomstige collega's variëren in leeftijd van begin 20 tot begin 50, van moeder van 2 kinderen tot eeuwige vrijgezel.*
 - Even though this age group seems rather big, it is excluding the sensitive group of people above 50 which are mainly target of ageism.

Indirect discrimination

Indirect discrimination is occurring when the sentence is implying that a young candidate is wanted without saying the age or using the word "jong".

Indirect discrimination is a bit more subtle than direct discrimination and occurs e.g. when job advertisements are asking for **students**. Even though everybody can be a student, students are usually young. By using this term it is implied that

the applicant is expected to be young.

Other examples than the student examples are:

- Talking about "de eerste/tweede stap in jouw carrière"
- Advertising a "bijbaan naast your opleiding" or "jouw eerste baan"
- addressing recent graduates ("net afgestudeerd", "schoolverlater")
- "starter op de arbeidsmarkt" or starter in general
 - Talking about junior and senior positions is not discriminating for this annotation task. Those labels are considered to be positions rather than age categories. In case you feel the term "junior" or "senior" is used more as an age label than a position, please add the `un_sure` label to the `no-discrimination` label.
- Giving **maximum of years** as experience
 - Giving minimum years of experience is considered to be a qualification assessment, the maximum amount of years is, however, age discriminating

Example sentences of indirect discrimination are:

- *Heb je net een technische opleiding afgerond op MBO/MTS niveau en ben je toe aan **je eerste échte baan?***
- *Deze functie is ideaal voor **studenten die naast hun studie een bijbaan zoeken.***
- *Deze **bijbaan past perfect naast je studierooster** omdat je iedere week kunt bepalen op welke dagen en tijden je de week erop wilt werken.*

Advertising **bijbaan** is not discriminating, only if it is advertised as a bijbaan for students (naast jouw studie etc.). But that often co-occurs

- *En beschik jij over **maximaal 5 jaar relevante werkervaring***
- *Voor onze opdrachtgever zijn wij op zoek naar een representatieve **schoolverlater (of met afgeronde MBO)** welke per direct beschikbaar is voor de functie van magazijnmedewerker.*
- *Om deze lijn de komende jaren door te zetten zijn we op zoek naar een **talentvolle starter** die als inside sales medewerkers aan de slag gaan.*
- *Voor onze hotels zijn we op zoek naar een gemotiveerde, leergierige student die zich wil ontwikkelen in de HR afdeling.*

Indirect discrimination occurs whenever concepts and words are used which imply that the perfect applicant is young (e.g. student-related references, career-starter etc.) without using the words "jong" or the specific age group.

Other

You can think of this label as a collection of sentences which do not quite fit into the category of direct and indirect but have some discriminating flair to it which makes you hesitant to use `no discrimination`.

It can be applied to samples which give a specific justification/explanation for the age discrimination.

A justification can look like this:

Dit is een vacature voor jongeren van 18 tot 27 jaar in het kader van de 'Aanpak jeugdwerkloosheid'

Further, some sentences invite people of different age groups. See e.g. this sentence:

De organisatie bestaat uit ongeveer 40 medewerkers die bestaan uit zowel jonge talenten als zeer ervaren medewerkers.

Wij zoeken een ervaren kracht of een jonge schoolverlater om zelf intern op te leiden tot expediteur

Please apply the `others` label to similar cases.

Also, as discussed above, please also use it for cases where either the company or the team is described as "jong", but where the applicant is only indirectly called young.

- *Als Simulation Engineer ben je werkzaam in een **jong team** onder begeleiding van een Lead Engineer.*

No-discrimination

This label, as the name suggests, should be used to mark sentences which contain no age discrimination. The data you will be labeling is a random set of sentences extracted from job descriptions and therefore may also contain descriptions

of the candidate with regards to their skills, details of the company and the job. Sometimes job description will entail ages or the word "jong", as in this case:

Ruime ervaring met diagnostiek en behandeling van kinderen en jongeren met een psychiatrische stoornis

Obviously, the term "jongeren" is not addressing the applicant but used to describe the job. It is not used in a discriminatory context.

If you are not sure if a sentence is really not discriminating, please add the `unsure` label to it. You can use it in combination with `no-discrimination` to indicate that you think it is most likely non-discriminatory.

Internship

Some job advertisements are internship-vacancies. Please label these samples with `internship`. This entails all samples where it is clear that an internship position is advertised. But this does not entail sentences in which an internship is mentioned as a requirement.

An example of a sentence advertising an internship:

Een dynamische stageplaats in een jong en informeel team

Please also use this label for **Traineeships**.

Not sure

As already mentioned in the explanations of the other labels, the `unsure` label can be applied whenever you are unsure about your judgement. In those cases, please apply it in combination with any of the other tokens.

In case you have no idea which category to apply, please only select the `unsure` token. There might be samples which fit in no category specified.

If the sentence seems to be ambiguous, please apply all labels which could match and add the `unsure` tag as well.

However, if you are using the `unsure` label frequently, please contact me (anna@textmetrics.com). I expect the majority of the sentences to be unambiguous. So if you end up using the `unsure` token too often, I was not precise enough with my descriptions.

Further examples

Here is another overview of a few different examples for each direct and indirect discrimination:

Direct

jong

Je bent woonachtig in de omgeving van Apeldoorn en je past in een jong en informeel professioneel team.

Als Commercieel Binnendienst Medewerker ga jij een jong en enthousiast team versterken.

Voor deze functie zoeken wij een jonge enthousiaste medewerker met:

leeftijd

Gezien de huidige teamsamenstelling zoeken wij iemand in de leeftijd tot 45 jaar.

Je hebt affiniteit met het onderwijs en je bent in staat om jonge mensen in de leeftijd van 16 tot 22 jaar te inspireren en te begeleiden bij hun leerproces.

Wij bieden thuiswerk voor studenten in de leeftijd vanaf 18 jaar en ouder.

max

Indirect

baan

Heb je net een technische opleiding afgerond op MBO/MTS niveau en ben je toe aan je eerste échte baan?

bijbaan

Deze functie is ideaal voor studenten die naast hun studie een bijbaan zoeken.

Deze bijbaan past perfect naast je studierooster omdat je iedere week kunt bepalen op welke dagen en tijden je de week erop wilt werken.

ervaring

*En beschik jij over **maximaal** 5 jaar relevante werkervaring*

opleiding

je bent bezig met een HBO opleiding communicatie of volg je een iniversitaire studie communicatie en ben je op zoek naar een stage!

Selectie criteria: HAVO 5 met diploma en Nederlands / wiskunde / Engels op niveau 3F, maximaal 22 jaar oud.

oud

Gezien de samenstelling van ons team zoeken we iemand die wat ouder is en ook al ervaring heeft.

Gezien de functie en de samenstelling van het team gaat de voorkeur uit naar een kandidaat die niet ouder is dan 21 jaar.

tussen

'Houd je ervan de gasten in de watten te leggen, ben je flexibel, leergierig, tussen de 20 en 40 jaar oud, enthousiast, en heb je passie voor ons vak, bell!'

min

Wat we van je verwachten: je bent een man van minimaal 21 jaar oud.

schoolverlater

Voor onze opdrachtgever zijn wij op zoek naar een representatieve schoolverlater (of met afgeronde MBO) welke per direct beschikbaar is voor de functie van magazijnmedewerker.

stap

Wil jij graag je eerste stappen zetten in jouw carrière?

start

Dat maakt twee perfecte redenen om je loopbaan bij ons te starten.

starte

Om deze lijn de komende jaren door te zetten zijn we op zoek naar een talentvolle starter die als inside sales medewerkers aan de slag gaan.

stud

Voor onze hotels zijn we op zoek naar een gemotiveerde, leergierige student die zich wil ontwikkelen in de HR afdeling.

Other

Als Simulation Engineer ben je werkzaam in een jong team onder begeleiding van een Lead Engineer.

De organisatie bestaat uit ongeveer 40 medewerkers die bestaan uit zowel jonge talenten als zeer ervaren medewerkers.

Dit is een vacature voor jongeren van 18 tot 27 jaar in het kader van de 'Aanpak jeugdwerkloosheid'

Internship

Vanaf februari 2015 komen er weer enkele stageplaatsen vrij voor 3de jaars stagiaires HBO- MWD, voor 10 maanden.

INTERNATIONALE COMMERCIELE TRAINEES (m/v) die een uitdagende start will maken in hun (internationale) carrière.

My Thesis at Textmetrics

For my Master's thesis I researching if AI tools can be used to build an age discrimination detector. I hope to find a way to identify reliably if a sentence of a job description discriminates based on age. To do so, I will need a set of sentences with human judgments which I can test my detector with. So I will use the data you label to assess how well my approach works.

Appendix B

Generated Regexes

```
toe\saan\sje\seerste\s?[a-z]*
toe\saan\sje\seerste\s?[a-z]*
[a-z]*\s?(\s?hbo\s?|\s?jaars\s?)\s?|\s?naar\s?)\s?(\s?student\s?|\s?(\s?jongeren\s?|\s?studerend\s?)\s?|\s?(\s?\ \s?|\s?jaars\s?)\s?)\s?)
[a-z]*\s?(\s?afgestudeerden\s?|\s?jong\-\professionals\s?)\s?|\s?(\s?carrière\s?|\s?startende\s?)\s?|\s?hbo\s?'ers\s?)\s?)
[a-z]*\s?[a-z]*\s?(\s?(\s?hbo\s?'ers\s?|\s?jong\-\professionals\s?)\s?|\s?(\s?carrière\s?|\s?afgestudeerden\s?)\s?)\s?|\s?startende\s?)
[a-z]*\s?studie\ste\scombineren\sis
jij\svolgt\sop\sdit\smoment\sde\sstudie
wij\szijn\seen\sjonge\sorganisatie\smet\snet\safgestudeerden\sen\sstuderende
[a-z]*\s?zijn\svij\sop\szoek\snaar\sjou
je\sbent\sop\szoek\snaar\seen\s2e\sstap\sin\sje\s?carrière
je\sbent\sop\szoek\snaar\seen\stweede\sof\sderde\sstap\sin\sje\s?carrière
[a-z]*\s?student\s?of\safgestudeerde\syoung\sprofessional
[a-z]*\s?het\suitvoeren\svan\seen\s?opdracht\project\s?in\slijn\smet\sje\sstudie
[a-z]*\s?baan\svoor\siedereen\adie\snaar\sschool\sgaat\sof\sstudeert
ben\sijj\seen\sstudent\s?[a-z]*
ben\sijj\siemand\sdie\s?[a-z]*
ben\sijj\sop\szoek\seen\stweede\sstap\sin\sje\s?carri
het\sis\sde\seerste\sof\stweede\s?carrière\s?voor\siemand
[a-z]*\s?tweede\sstap
[a-z]*\s?(\s?(\s?met\s?|\s?na\s?)\s?|\s?van\s?)\s?je\s?(\s?praktijkstage\s?|\s?(\s?opleiding\s?|\s?studie\s?)\s?)
[a-z]*\s?[a-z]*\s?of\s?(\s?(\s?2e\s?|\s?(\s?student\s?|\s?(\s?4e\s?|\s?tweede\s?)\s?)\s?)\s?|\s?derde\s?)\s?|\s?(\s?als\s?|\s?(\s?net\s?|\s?volg\s?)\s?)\s?)\s?[a-z]*
[a-z]*\s?(\s?(\s?is\s?|\s?bent\s?)\s?|\s?voorkeur\s?)\s?(\s?student\s?|\s?nog\s?)
[a-z]*\s?(\s?wachten\s?|\s?startende\s?)\s?\ \s?[a-z]*
studenten\s?(\s?(\s?onder\s?|\s?de\s?)\s?)\s?|\s?studerend\s?)\s?[a-z]*
[a-z]*\s?de\s?(\s?omgeving\s?|\s?vachturen\s?)
(\s?volgt\s?|\s?je\s?)\s?[a-z]*\s?(\s?(\s?mbo\s?|\s?student\s?)\s?|\s?(\s?dit\s?|\s?studerend\s?)\s?|\s?(\s?studeert\s?|\s?universitair\s?)\s?)\s?)
(\s?(\s?wij\s?|\s?/\s?)\s?|\s?(\s?(\s?(\s?nog\s?|\s?1\s?)\s?)\s?|\s?starter\s?)\s?|\s?tenzij\s?)\s?|\s?(\s?voorkeur\s?|\s?student(en)\s?)\s?)\s?|\s?ben\s?)\s?)\s?|\s?studeer\s?)
\s?[a-z]*\s?(\s?(\s?(\s?student\s?|\s?(\s?tenzij\s?|\s?gezocht\s?|\s?studerende\s?)\s?)\s?)\s?|\s?(\s?jouw\s?|\s?(\s?zoekt\s?|\s?recent\s?)\s?)\s?)\s?|\s?nog\s?)
```

Bibliography

- [1] A. Fokkens and C. J. Beukeboom. Leeftijdscriminatie in vacatureteksten: Een herhaalde geautomatiseerde inhoudsanalyse naar verboden leeftijd-gerelateerd taalgebruik in vacatureteksten uit 2017 en 2019. Technical report, het College voor de Rechten van de Mens, 2020.
- [2] Robert N Butler. Age-Isms: Another Form of Bigotry. *The Gerontologist*, 9(4 Part 1):243–246, 12 1969.
- [3] Bill Bytheway. Ageism and Age Categorization. *Journal of Social Issues*, 61(2):361–374, 6 2005.
- [4] Iris Andriessen, Henk Fernee, and Karin Wittebrood. Ervaren discriminatie in Nederland. Technical report, Sociaal en Cultureel Planbureau, 2014.
- [5] De juiste persoon op de juiste plaats - De rol van stereotypering bij de toegang tot de arbeidsmarkt. Technical report, College voor de Rechten van de Mens, 2013.
- [6] Peggy Voss, Ehud Bodner, and Klaus Rothermund. Ageism: The Relationship between Age Stereotypes and Age Discrimination. pages 11–31. Springer, Cham, 2018.
- [7] Tracey L. Gendron, E. Ayn Welleford, Jennifer Inker, and John T. White. The Language of Ageism: Why We Need to Use Words Carefully. *The Gerontologist*, 56(6):997–1006, 12 2016.
- [8] John Macnico. Ageism and Age Discrimination Some Analytical Issues (Web Report).
- [9] H. Jonson. We Will Be Different! Ageism and the Temporal Construction of Old Age. *The Gerontologist*, 53(2):198–204, 4 2013.
- [10] Malcolm Sargeant. *Age Discrimination in Employment*.
- [11] AS Fokkens, CJ Beukeboom, and E Maks. Leeftijdscriminatie in vacatureteksten: Een geautomatiseerde inhoudsanalyse naar verboden leeftijd-gerelateerd taalgebruik in vacatureteksten: Rapport in. 2018.
- [12] Ian Burn, Patrick Button, Luis Felipe Munguia Corella, and David Neumark. Older Workers Need Not Apply? Ageist Language in Job Ads and Age Discrimination in Hiring. Technical report, National Bureau of Economic Research, Cambridge, MA, 12 2019.
- [13] Panggih Kusuma Ningrum, Tatdow Pansombut, and Attachai Ueranantasun. Text mining of online job advertisements to identify direct discrimination during job hunting process: A case study in Indonesia. *PLOS ONE*, 15(6):e0233746, 2020.
- [14] Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. doccano: Text Annotation Tool for Human, 2018.
- [15] Mary L. McHugh. Interrater reliability: The kappa statistic. *Biochemia Medica*, 22(3):276–282, 2012.

- [16] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.
- [17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *arXiv preprint arXiv:1310.4546*, pages 1–9, 10 2013.
- [18] Matthew E Peters, Mark Neumann, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. Technical report.
- [19] Andrew M. Dai and Quoc V. Le. Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems*, 2015-January:3079–3087, 11 2015.
- [20] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A Primer in BERTology: What we know about how BERT works. *arXiv*, 2 2020.
- [21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova Google, and A I Language. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Technical report.
- [22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- [23] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut, and Google Research. ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS. Technical report.
- [24] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98, 7 2019.
- [25] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 10 2020. Association for Computational Linguistics.
- [26] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.
- [27] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to Use t-SNE Effectively. *Distill*, 1(10), 10 2016.
- [28] Shusen Liu, Peer-Timo Bremer, Jayaraman J. Thiagarajan, Vivek Srikumar, Bei Wang, Yarden Livnat, and Valerio Pascucci. Visual Exploration of Semantic Relationships in Neural Word Embeddings. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):553–562, 1 2018.
- [29] Andreas Holzinger. From Machine Learning to Explainable AI. In *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, pages 55–66. IEEE, 8 2018.
- [30] Apa) ; Marrero and M Urbano. A Semi-Automatic and Low-Cost Method to Learn Patterns for Named Entity Recognition. *Natural Language Engineering*, 24(1):39–75, 2018.
- [31] Rohit Babbar and Nidhi Singh. *Clustering Based Approach to Learning Regular Expressions over Large Alphabet for Noisy Unstructured Text*. 2010.

- [32] Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Eric Medvet, and Enrico Sorio. Automatic Synthesis of Regular Expressions from Examples. *Computer*, pages 1–1, 2013.
- [33] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Learning Text Patterns Using Separate-and-Conquer Genetic Programming. pages 16–27. 2015.
- [34] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Inference of Regular Expressions for Text Extraction from Examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230, 5 2016.
- [35] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Can a Machine Replace Humans in Building Regular Expressions? A Case Study. *IEEE Intelligent Systems*, 31(6):15–21, 11 2016.
- [36] Peter J Angeline. Genetic programming: On the programming of computers by means of natural selection,. *Biosystems*, 33(1):69–73, 1 1994.
- [37] Stuart J (Stuart Jonathan) Russell and Peter Norvig. *Artificial intelligence : a modern approach LK* - <https://ru.on.worldcat.org/oclc/1124776132>. Pearson, Hoboken, NJ SE - xvii, 1115 pages : illustrations (some color) ; 27 cm., fourth edi edition, 2021.
- [38] B Craenen, A Eiben, and E Marchiori. How to Handle Constraints with Evolutionary Algorithms. In *The Practical Handbook of Genetic Algorithms*. Chapman and Hall/CRC, 12 2000.
- [39] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Active Learning of Regular Expressions for Entity Extraction. *IEEE Transactions on Cybernetics*, 48(3):1067–1080, 3 2018.
- [40] Burr Settles. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 6 2012.