Universität Zürich[UZH]

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Arts
der Philosophischen Fakultät der Universität Zürich

Recognition of ICT Terms in German Job Ads: A Case Study in Bootstrapping a BERT-based NER model

Verfasserin/Verfasser: Bühlmann Eva

Matrikel-Nr: 05-058-912

Referentin/Referent: Dr. Simon Clematide

Institut für Computerlinguistik

Abgabedatum: 01.06.2021

Abstract

Job ads provide unique data for monitoring the labour market. As part of the NRP77 project *Digital Transformation*, this thesis aims to detect ICT terms in German-language Swiss job ads. No pre-trained models for recognizing this special domain entity type is available in German. Therefore, the recognition of ICT terms is approached by bootstrapping a NER model based on domain-specific BERT embeddings, making use of transfer learning. In a multi-step approach a NER model was trained and iteratively improved by adding training data through correcting the models predictions on new samples. A targeted sampling strategy based on a topic model was used to cover a wide variety of ads and ICT terms in the training data. Starting with a training corpus with only 90 manually annotated ads, the successive addition of 500 new ads significantly improved performance, with the final model achieving an F-score of 90.42. This is an improvement of 1.38 percentage points over a model with generic BERT embeddings, showing the power of domain-specific embeddings. The inclusion of term lists in the training data, however, has not shown a benefit for model performance.

Zusammenfassung

Stellenanzeigen liefern einzigartige Daten zur Beobachtung des Arbeitsmarktes. Als Teil des NFP77-Projekts *Digitale Transformation* zielt diese Arbeit darauf ab, IT-Begriffe in deutschsprachigen Schweizer Stellenanzeigen zu erkennen. Vortrainierte Modelle für die Erkennung dieses spezifischen Entitätstyps in deutscher Sprache gibt es keine. Daher wird die Erkennung von IT-Begriffen durch Bootstrapping eines NER-Modells unter Verwendung von Transfer Learning, basierend auf domänenspezifischen BERT-Embeddings angegangen. In einem mehrstufigen Ansatz wird ein NER-Modell trainiert und iterativ durch Hinzufügen von Trainingsdaten verbessert, indem die Vorhersagen des Modells auf neuen Stichproben korrigiert wurden. Eine gezielte Sampling-Strategie, die auf einem Topic-Modell basiert, erlaubt es, eine möglichst grosse Vielfalt an Anzeigen und IT-Begriffen in den Trainingsdaten abzudecken. Ausgehend von einem Trainingskorpus mit nur 90 manuell annotierten Anzeigen, konnte das sukzessive Hinzufügung von 500 neuen Anzeigen die Leistung deutlich verbessern, wobei das endgültige Modell einen F-Score von 90,42 erreichte. Dies ist eine Verbesserung von 1,38 Prozentpunkten gegenüber einem Modell mit generischen BERT Embeddings, was die Leistungsfähigkeit von domänenspezifischen Embeddings zeigt. Der Einbezug von Termlisten in die Trainingsdaten hatte hingegen keinen positiven Einfluss auf die Resultate.

Acknowledgement

I thank Dr. Simon Clematide for his continuous support, his highly valuable suggestions and feedback during the development of this work. I would also like to thank Ann-Sophie Gnehm for her always very helpful inputs and insights. Additionally, I thank the project team for providing me with various materials and resources that were a very important foundation for my work.

Contents

List of Figures

List of Tables

List of Acronyms

BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
ICT	Information and Communication Technology
JSONL	JavaScript Object Notation (Json) Lines format
LDA	Latent Dirichlet Allocation
LSTMs	Long short-term memory Networks
NER	Named Entity Recognition
NLP	Natural Language Processing
NRP77	National Research Programme "Digital Transformation"
POS	Part-Of-Speech
SJMM	Swiss Job Market Monitor
XML	eXtensible Markup Language

1 Introduction

1.1 Motivation

The NRP77 project *Digital Transformation*[1] aims to analyze how digitization is changing task and qualification profiles of employees in Switzerland. By applying advanced text mining techniques to a large and multilingual corpora of job ads, tasks and required qualifications used at workplaces will be identified for the period from 1990 to the present. In the context of this project, the goal of this thesis is to detect Information and Communication Technology (ICT) terms in the German job ad corpora.

Job ads provide unique data about employers' staff needs at the job level (Gnehm and Clematide [2020]). Therefore, Text mining on job ads has taken an important role in monitoring the requirements of the labor market (Atalay et al. [2020], Gnehm and Clematide [2020], Dawson et al. [2019], Boselli et al. [2018], Karakatsanis et al. [2017]) with some recent studies focusing particularly on the impact of digitization (Fareri et al. [2020], Pejic-Bach et al. [2020]).

The recognition of ICT terms, aimed at by this work, can serve to extract ICT skills and tasks and should help to build an indicator of digitization of the Swiss job market, over the time and different professions or industries.

There are different approaches for finding ICT terms in job advertisements. Some initial experiments with a phrase matching approach were not very successful, having problems with recognizing multi word terms. At the same time, first attempts with a BERT based Named Entity Recognition (NER) model showed very promising results. Therefore, the initial approach was not pursued further, but the focus was put entirely on bootstrapping a NER model for ICT terms, using transfer learning through BERT based language models. The `terms.teach` recipe from *prodigy* offers a simple way to quickly expand existing term lists. However, the original idea of using these lists to annotate new data via pattern matching was not pursued, because

[1] see `http://p3.snf.ch/Project-187333`

it resulted in a lot of false positive matches, which made the process very inefficient. Nevertheless, it will be investigated whether the term lists can be useful by adding them to the training data.

1.2 Research Questions

The research questions to be answered in this thesis, are:

1. How can the problem of recognizing ICT terms in job ads be formalized and conceptualized in terms of annotation guidelines?
    a) To what extent is it possible to solve the recognition of ICT terms as a NER tagging problem with a flat approach and how much simplification is needed for this?
    b) What are the challenges of creating annotation guidelines for a new, special domain entity type (ICT terms in job ads)?
2. How much and which kind of training data is needed to quickly bootstrap a BERT based NER model for ICT terms?
    a) How can appropriate examples be sampled for annotation?
    b) How much training data is needed to reach an F-Score of 90%?
    c) Can the NER model benefit from term lists?
    d) What is the benefit of using domain-specific BERT embeddings in comparison to general language embeddings?

1.3 Thesis Structure

In Chapter 2 the used material and data preparation process, including the development of annotation guidelines, will be described. Chapter 3 then gives an overview of the underlying methods, this involves Text embeddings, the problem of NER, the used *prodigy* annotation tool and topic modelling. In Chapter 4, the performed experiments and results are described, with a main focus on the iterative process of training the NER model and increasing training data. Chapter 5 will compare and discuss the results, before a final conclusion is given in Chapter 6. The scripts developed in the course of this work are available at `https://github.com/evabuehlmann/ICTterm-recognition`.

Note that examples of ICT terms and excerpt from the ads are presented in their original wording in German and without additional translation. Therefore, it is advantageous if the reader of this thesis understands some German.

2 Materials and data preparation

2.1 Description of Job Ad Corpus

The provided job ad corpus consists of Swiss job advertisements from different sources. It is based on the Swiss Job Market Monitor (**SJMM**), which consists of job ads in German, French and Italian, representative for the Swiss job market since 1950 and covering all advertising channels of relevance during the observation period (SMM [2021]). This corpus is complemented by an online-only corpus, with ads from 2015-2020, referred to as **SJMM-O** (SJMM-Online) and another large online ad corpus crawled by a private company, referred to as **OA** (Online Ads), which contains ads from 2014-2018. Both online corpora include ads in German, French, English and Italian. Since there are many duplicates, especially in the OA corpus, deduplication was performed based on ad IDs and the ad text. This leaves a total of over 2.5 million ads.

The present thesis focus on German ads only. Furthermore, for Bootstrapping the NER model the selection of training data was limited to the time period from 2001 to 2020, as ICT terms are expected to occur predominantly in these recent years. Follow-up work will explore, how the terms recognition works for older material. Table 1 shows the number of deduplicated German ads per year and source for the time selected time period.

2.2 Sampling Training Data

In total, four samples with different content and retrieval focuses were selected from the entire job ad corpus as training data for the NER model. While the first sample was selected using keywords, the other three samples were chosen based on a topic model (see Chapter 3.4).

When training a NER model, it is crucial to have training data, that covers the type of entities that one is most interested in (Honnibal [2017]). Therefore, the

Year	SJMM	SJMM-O	OA	Total
2020	0	6,995	0	6,995
2019	2,815	11,995	0	14,810
2018	2,678	12,064	414,108	428,850
2017	2,584	13,187	477,895	493,666
2016	25,99	11,989	442,135	456,723
2015	2,491	3,500	366,604	372,595
2014	2,930	0	324,711	327,641
2013	2,996	0	0	2,996
2012	3,007	0	0	3,007
2011	2,947	0	0	2,947
2010	2,898	0	0	2,898
2009	1,433	0	0	1,433
2008	1,481	0	0	1,481
2007	1,395	0	0	1,395
2006	1,343	0	0	1,343
2005	1,091	0	0	1,091
2004	1,051	0	0	1,051
2003	1,067	0	0	1,067
2002	1,159	0	0	1,159
2001	1,354	0	0	1,354
Total	**39,319**	**59,730**	**2,025,453**	**2,124,502**

Table 1: Statistics German Job Ad Corpus 2001-2020

samples were selected with different content focuses, in order to provide the NER model with various types of ads and ICT terms. At the beginning a strongly recall-oriented approach was followed, by sampling ICT-rich ads in order to expose the model to many different examples of ICT terms. Through the later inclusion of samples with less ICT-oriented ads, it is ensured that the final model can not only handle ads from the ICT domain, but also learns to correctly annotate ads that contain no ICT terms at all, or only a few general or very sector-specific ICT terms. The iterative process of training the NER model and sampling new training data is described in Chapter 4.2.

As shown in Table 1 the number of ads from the OA corpus is significantly higher than for the other two sources. Therefore, to prevent training data to consist almost exclusively of OA ads, the samples are not drawn purely at random from the entire

corpus, but care is taken to ensure that ads from all sources and years are considered. The exact composition of the different samples by year and source can be seen in the Tables in appendix A.

In addition, not the entire ads were used as training data, but only segments associated with certain text zones, as described in the next Chapter.

2.2.1 Zoned Job Ads

As a preparatory work for precise skill and task extraction Gnehm and Clematide [2020] have performed text zoning on the Swiss job ad corpus, as a token-level sequence labeling task and by using contextualized embeddings. 8 different zones have been distinguished:

- z10: Company Description
- z20: Reason of Vacancy
- z30: Administration/Residual Text
- z40: Job Agency Description
- z50: Material Incentives
- z60: Job Description
- z70: Required Hard Skills
- z80: Required Personality/Soft Skills

ICT terms are most prominent in job descriptions and required hard skills. Thus, when sampling ads for the present study, the focus was put on text segments belonging to zones 60 and 70. However, since the zones are on token level and zones can change within a sentence – even several times (see example below), not only the tokens belonging exactly to these zones were selected, but also surrounding text. A threshold of +/- 10 tokens has been defined, in order to obtain sufficient context.

Example of nested text zones in one sentence

Zur Verstärkung [z20] unseres bestehenden Python-Entwicklerteams [z60] suchen wir [z30] eine(n) motivierte(n) [z80] und erfahrene(n) [z70] Python-Entwickler(in). [z60]

2.2.2 Extracting Job Ads from XML files

The job ad data was provided in XML format, containing meta-information and information on token-level (including "space tokens") for every ad, including POS-tag, lemma, position and zone – but no information on sentence segmentation. It is difficult to identify sentence boundaries in the ads because they contain many lists, and punctuation is not always reliable.

Below is an excerpt from such a zoned-job ad in XML format:

```
<?xml version='1.0' encoding='UTF-8'?>
<ads>
  <ad channel="9" id="5479159" year="2014">
   <text_annotated language="de" length="1381">
      ...
      <token lemma="zu" pos_tag="APPRART" position="103" zone="20">Zur</token>
      <space lemma=" " pos_tag="WS" position="104" zone="20"> </space>
      <token lemma="Verstärkung" pos_tag="NN" position="105" zone="20"
        >Verstärkung</token>
      <space lemma=" " pos_tag="WS" position="106" zone="20"> </space>
      <token lemma="unser" pos_tag="PPOSAT" position="107" zone="60">unseres</token>
      <space lemma=" " pos_tag="WS" position="108" zone="60"> </space>
      <token lemma="bestehend" pos_tag="ADJA" position="109" zone="60"
        >bestehenden</token>
      <space lemma=" " pos_tag="WS" position="110" zone="60"> </space>
      <token lemma="Python-Entwicklerteams" pos_tag="NN" position="111" zone="60"
        >Python-Entwicklerteams</token>
      <space lemma=" " pos_tag="WS" position="112" zone="60"> </space>
      <token lemma="suchen" pos_tag="VVFIN" position="113" zone="30">suchen</token>
      <space lemma=" " pos_tag="WS" position="114" zone="30"> </space>
      ...
   </text_annotated>
</ad>
</ads>
```

A Python script[1] was used to extract the relevant segments from the sampled job ads in XML-format. As described before, for every ad, tokens and spaces belonging to the text zones 60 and 70 and +/- 10 surrounding tokens and spaces were extracted. The extracted text segments for each ad were then merged again into one text, without considering sentence boundaries.

[1] `extract_topic-based-samples.py` for topic-based samples
`extract_sample-0.py` for the ICT-keyword based sample
Available at: `https://github.com/evabuehlmann/ICTterm-recognition`

The extracted texts, together with a unique ad-Id and meta-information, were stored in a JSONL-File, which could later be directly used as input for the annotation with `prodigy ner.correct` (see Chapter 3.3.2).

The following Figures shows an example of an entire ad, before (Figure 1) and after (Figure 2) the extraction based on text zones. It can be seen that only the tokens belonging to the zones 60 and 70 and its surroundings are kept. Since sentence boundaries are not considered, it can happen that segments are cut or interrupted in the middle of sentences and merged with other segments.

\nDank langjähriger Erfahrung und erfolgreicher Umsetzung von tausenden Projekten geniesst unsere Kundin einen sehr guten Ruf als branchenübergreiffendes Softwareunternehmen. Dahinter steckt viel Engagement und Innovationsgeist der Mitarbeitenden sowie auch die Unterstützung des Arbeitgebers bezüglich Weiterbildungsmöglichkeiten. Diese Kombination hat sich bewährt. [z10]Für den weiteren Ausbau [z20]am Standort in Zürich [z10]suchen wir [z30]einen\nSoftware Engineer C# / replaced-dns [z60](m/w)\n[z80]Region Zürich/Schaffhausen\n\nIhre Herausforderung\n\nIn einem dynamischen Team entwickeln und realisieren Sie komplexe Software-Lösungen für Kunden - innovativ, bedürfnisgerecht und individuell. Dabei verwenden Sie die modernsten Technologien und Methoden. Ihre Hauptaufgabe ist die objektorientierte Software-Entwicklung mit C# / replaced-dns. [z60]Sie schätzen projektorientierte und methodische Arbeitsweisen und haben den Willen [z80]ein Projekt von A-Z umzusetzen. Die Projekte sind sehr vielfältig und in unterschiedlichen Branchen angesiedelt.\n\n[z60]Ihre Qualitäten und Begabungen\n\n Studium in Informatik (FH/Uni/ETH)\n Erfahrung in der objektorientierten Programmierung in C# / replaced-dns\n [z70] Qualitätsbewusstsein bezüglich hochwertigem Code\n Analytisches Denkvermögen, lösungsorientierte Persönlichkeit\n [z80]Sehr gute Deutschkenntnisse und gute Englischkenntnisse\n\n[z70]Das Unternehmen legt viel Wert auf die Qualifikation ihrer Mitarbeitenden [z10]und unterstützt Weiterbildungen[z50]. [z10]Spricht Sie eine [z30]Herausforderung [z60]in einem technologisch breit aufgestellten Umfeld an? [z10]Dann freut sich Ruben Ehrismann auf Ihre Bewerbungsunterlagen per E-Mail oder Ihren Anruf.\n[z30]Consult & Pepper AG\n[z40]Ruben Ehrismann\nConsultant\nTelefon: 052 369 20 20\nreplaced-email\n\nreplaced-dns\n\nBern • Luzern • St. Gallen • Winterthur\n\n\n\n\n[z30]

Figure 1: Example of entire ad with text zones

Standort in Zürich [z10]suchen wir [z30]einen\nSoftware Engineer C# / replaced-dns [z60](m/w)\n[z80]Region Zürich/Schaffhausen\n\nIhre Herausforderung\n\nIn einem dynamischen Team entwickeln und realisieren Sie komplexe Software-Lösungen für Kunden - innovativ, bedürfnisgerecht und individuell. Dabei verwenden Sie die modernsten Technologien und Methoden. Ihre Hauptaufgabe ist die objektorientierte Software-Entwicklung mit C# / replaced-dns. [z60]Sie schätzen projektorientierte und methodische Arbeitsweisen und haben den Willen [z80]ein Projekt von A-Z umzusetzen. Die Projekte sind sehr vielfältig und in unterschiedlichen Branchen angesiedelt.\n\n[z60]Ihre Qualitäten und Begabungen\n\n Studium in Informatik (FH/Uni/ETH)\n Erfahrung in der objektorientierten Programmierung in C# / replaced-dns\n [z70] Qualitätsbewusstsein bezüglich hochwertigem Code\n , lösungsorientierte Persönlichkeit\n [z80]Sehr gute Deutschkenntnisse und gute Englischkenntnisse\n\n[z70]Das Unternehmen legt viel Wert [Z10] Weiterbildungen[z50]. [z10]Spricht Sie eine [z30]Herausforderung [z60]in einem technologisch breit aufgestellten [z10]

Figure 2: Example of extracted ad ads with text zones

2.3 ICT Term Lists

Two lists containing ICT terms were available as part of the project: One list[2] containing 3,306 ICT terms and a shorter list[3] with 291 terms. Both lists have been created by domain experts, partly resulting from exploration of the job ads, partly based on external resources (e.g. Dierdorff et al. [2006]). These lists built the basis for expanding ICT terms with *prodigy* `terms.teach` (see Chapter 3.3.1) and for extracting a first sample of ads containing ICT keywords. This sample was annotated manually (see Chapter 2.4.2) and was used to train the initial NER model as described in Chapter 4.2.3.

2.4 Manual Gold Standard

Since no training material with annotated ICT terms was available, a small corpus of manually annotated ads was created as a first step. This sample also helped to define and refine formal annotation rules.

For a general introduction into manual annotation for machine learning models see Pustejovsky and Stubbs [2012]. With the "MATTER model" they describe an annotation development cycle, suggesting an iterative approach. MATTER stands for **M**odel, **A**nnotate, **T**rain, **T**est, **E**valuate, **R**evise. Its core idea is to first model the annotation task, by defining the annotation goal, the relevant resources and specifying annotation standards. This allows to create an annotated gold standard corpus, which can be used to train a first model. After testing and evaluating this model, a revision of the previous steps should be made. For example, it may prove useful to expand the domain of the corpus or revise unclear annotation specifications. The MATTER model was applied in this work in that the annotation rules were further specified in some unclear cases and the gold standard corpus was extended in several steps.

The following sections describe the annotation rules used for the ICT terms, including how they were created, the challenges involved, and how they were used to manually annotate an initial gold standard corpus. Selected extensions to the training corpus during the process are discussed in Chapter 4.2.

[2] `itterms_sjmm_9018.txt`

[3] `ittools_x28.xlsx`

2.4.1 Definition of Annotation Rules

Before starting the annotation process, it was essential to first determine what is to be considered an ICT term, in order to enable a consistent annotation. Due to the complexity and variety of potential ICT terms, an application-oriented annotation rule set with examples was determined, instead of just giving a definition. Since this decision is not only of a technical nature, but related to the goal of the overall project, which should benefit from this work, these rules were defined and discussed in exchange with the project team. As a basis for this discussion, the most common problematic and borderline cases and examples that appear in the first gold standard set (Sample-0) were collected and organized according to different aspects. Uncertainties that arose during annotating Sample-0 and in later annotation rounds (see Chapter 4.2) were used to further refine these rules. The resulting annotation rules are described in the next section.

2.4.1.1 Annotation Rules for ICT Terms

Additions to ICT terms and composites
Descriptive additions to ICT terms, like *"Knowledge in ..."* or *"users of ..."* that are not composites are not counted as part of the term. Example:

- *Sie kennen sich mit den Programmen der [[Office 2010-Palette]] aus und verfügen über ausgezeichnete Kenntnisse in [[Excel]]*

If, on the other hand, such expressions come as composites with or without hyphen, they are regarded as one ICT term. This also includes expressions which are not explicitly written as composites but which could be converted into a compound by simply merging their individual components[4]. Examples:

- *Sie kennen sich mit den [[MS-Office Programmen]] aus und bringen sehr gute [[Excel-Kenntnisse]] mit*
- *Für ein [[e-Commerce Projekt]] suchen wir Unterstützung im [[Front-End Bereich]]*

In cases where an ICT term is used as an adjective, the associated noun or nominal phrase is generally added to the ICT term. Conversely, adjectives that further describe an ICT term but do not themselves have an ICT meaning are not considered part of the term. Examples:

[4]German spelling is often very inconsistent with regard to hyphenation and concatenation, especially with technical terms.

- *Sie verfügen über Erfahrungen in der Entwicklung und Umsetzung von durchgängigen, [[IT-basierten Prozessketten]]*
- *Mit Ihrer Tätigkeit sind Sie massgebend an der Gestaltung und dem Fortschritt der [[digitalen Zukunft]] des Luzerner Kantonsspitals beteiligt*
- *Junior Controller 100% (w/m) mit sehr guten [[Excel-Kenntnissen]]*

Job titles and completed education

Job titles, which are often used as the title of the ad (together with the desired gender and job percentages) are considered as one connected ICT term. Formal designations of training qualifications or titles which appear directly after the job title are also added to the ICT term. Examples:

- *[[Senior System Engineer Linux]] 100% (w/m)*
- *Für diese vielseitige, anspruchsvolle Tätigkeit verfügen Sie über eine Ausbildung als [[Informatik Techniker HF/FH]] oder gleichwertig*

However, in cases where the required education is mentioned but not directly attached to the job title, the education is not counted towards the term. Examples:

- *Erfolgreich abgeschlossene [[Informatik Ausbildung]] ([[Software Entwickler]] , Abschluss FH, HF, UNI oder vergleichbares)*
- *Höhere Fachausbildung (Ing. HTL / FH / ETH , [[Wirtschaftsinformatiker / in]] oder TS*

More general descriptions of experience or educational background, are also not included in the ICT term. Examples:

- *Lehre als [[Informatiker / in]] oder gleichwertig*
- *Wir erwarten von Ihnen eine abgeschlossene Ausbildung als [[Informatikerin]]*

Gender-specific notations

Gender-specific spellings or endings of job titles are combined into one ICT term. Examples:

- *[[Software Entwickler/-in]]*
- *Das Bundesamt für Umwelt BAFU sucht [[ICT-Beraterin/Berater]]*

Notations of the searched gender for the position, typically in brackets after the job title, by contrast, are not considered part of the ICT term. Example:

- *Suchen wir Sie als [[System Engineer]] (w/m)*

Proper names and other descriptive names

For company names or other fixed names (e.g. name of a department), components consisting of ICT terms are extracted separately instead of annotating the entire name as an ICT term. Example:

- *Für unser Team [[Informatik]] & Prozessdesign in Sachseln suchen wir*

Abbreviations

Another difficulty is dealing with abbreviations that are introduced together with the associated phrase. As a rule, the entire expression, including abbreviations, is annotated here as a coherent term. Example:

- *Zuständig für Betrieb , Unterhalt und Weiterentwicklung der [[Data Center Automatisierung]] mit [[Microsoft Service Manager SMA]] und [[MS PowerShell]]*

Enumerations

In the case of enumerations of similar ICT terms, often separated by slashes (e.g. *Windows 7 / Windows 8*), all terms are usually annotated separately. However, if elliptical omissions occur, where the individual components have no ICT meaning on their own (e.g. *Microsoft Windows Server 2003 / 2008 / 2012*, they are combined to avoid residual annotation, like [[2008]]). Example:

- *Sie bringen einige Jahre praktische Erfahrung aus folgenden Bereichen mit : [[Microsoft Windows]] [[Windows 7]] / [[Windows 8]] / [[Windows 10]] [[Microsoft Windows Server 2003 / 2008 / 2012]] [[Microsoft Exchange Server 2007 / 2012 / 2016]]*

In the case of enumerations of multiple ICT terms with elliptical compounds, initially a two-stage annotation procedure was chosen. On the one hand, the entire phrase is bracketed and, on the other hand, the individual components within the phrase are annotated separately as ICT terms. Examples:

- *Mitarbeit in der [[[Datenbank-]], [[Applikationsserver-]] und [[Unix-Administration]]]*
- *Fachliche Unterstützung der Kunden in laufenden Validierungsprojekten, insbesondere in der [[IT-Systemvalidierung]] (z.B. für [[[ERP-]], [[MES-]], [[LIMS-Systeme]]]*
- *Kategorie [[[System Administration] / [[-Engineering]] / [[-Architektur]]]*
- *Mehrjährige Praxiserfahrung im [[[1st]] und [[2nd Level ICT-Support]]]*

However, the effective recognition of such complex terms was not part of the present work. The later used *prodigy* annotation recipe and *spaCy* NER model only allow for single-level annotation, therefore the two-level annotation was not pursued further. Hence, for simplification, only the inner terms of multi-level terms were annotated. Nevertheless, the annotation rules described above may help for possible further experiments with the recognition of multi-level terms.

Ambiguous ICT terms, borderline cases

There are many cases where it is not clear whether a word has the character of an ICT term. Especially in job advertisements from the ICT sector, many words appear that have a different meaning in the ICT environment than in normal language usage. If a term clearly refers to a specific ICT aspect or task due to its context, it should be classified as such. Examples:

- *Weiterentwicklung und betreuen bestehender [[Schnittstellen]] , [[Websites]] und [[Webapplikationen]]*
- *Der Bereich Architektur und Methoden, Abteilung [[Informatik]], ist für das [[Architekturmanagement]] und die Bereitstellung [[integrierter Systeme]] verantwortlich*
- *Sie bringen mehrere Jahre Berufserfahrung in der Administration und dem Betrieb von [[SAP Systemen]] und [[Oracle Datenbanken]] auf [[HPUX]] / [[Linux]] in [[komplexen Umgebungen]] mit.*
- *Installation und Wartung von [[Arbeitsplatzsystemen]]*
- *Planung, Konzeption und Durchführung von [[Testaufgaben]] und [[Testfällen]]*

However, it should be prevented that very generic expressions, which in the context of ICT job task description can acquire some ICT character (e.g. *Konzeption, Entwicklung* or *Visualisierung*) are classified as ICT terms. Examples:

- *Für die Mitarbeit an anspruchsvollen Projekten erstellen Sie Visualisierungen von Konzepten bis zur Fertigstellung von [[High-End Renderings]].*
- *Entwerfen und umsetzen von Designs*

These rule, related to ambiguous ICT terms, was by far the most difficult to implement in the later annotation process. For example, should the terms *User support, Nutzer, Integration* or *Prozess* be annotated as ICT term if they are used in an ICT context, and what if the context is not clear? It is very difficult to formulate general rules, since new individual cases keep appearing. However, during the annotation process, I was able to identify and generalize a recurring type of uncertainty. It in-

volves rather general activity descriptions, which refer to another (ICT) term. This helped me to formulate an additional rule:

General activity words that may also have an ICT meaning and that refer to another ICT term are not annotated as ICT terms. Hence, in the following example, "Implementation" and "Integration" are not annotated, but "IT-Systeme" are. Example:

- *Hier sind Sie mit dabei, wenn wir Kunden rund im die Implementierung und Integration von neuen [[IT-Systemen]] in die bestehende [[IT-Anwendungslandschaft]] zur Seite stehen.*

Nevertheless, there is no clear truth for annotation in such borderline cases.Therefore, it was important to formulate as precise rules as possible that serve the overall project goal.

2.4.2 Creation of a Manual Gold Standard (Sample-0)

A sample of 90 ads extracted across the entire corpus served as the basis for first the manual gold standard. The requirement for this first gold standard (Sample-0) was, that it contained a large number of ICT terms, so that the model could learn to recognize them. Furthermore, this sample should also help to formulate annotation rules (see Chapter 2.4.1), which only makes sense if a high variety of potential ICT terms is included. A purely random selection would have resulted in a large number of ads without ICT terms. Therefore, at this step, a rather simple approach was chosen to ensure the inclusion of many different ICT terms.

Within the entire German job ad corpus, only ads containing at least one term from a list of ICT keywords were sampled[5]. As soon as an ad with a certain term was found, this term was removed from the keyword list in order to avoid selecting further ads based on the same keyword again, and thereby cover as large a variety of terms as possible. The used keyword list is based on the x28 term list (see Chapter 2.3), however, some very short and potentially ambiguous terms, such as "R", "Kat", "Basic" or "Migration" have been removed manually. The resulting ICT keyword list contained 258 single and multi-word terms.

Additionally, when sampling the ads, a regular distribution across the different sources (SJMM, SJMM-O, OA) and years was respected. For the exact distribution of sampled ads, see Table 14 in the appendix.

[5]see Python script `extract_sample-0.py`, available here: `https://github.com/evabuehlmann/ICTterm-recognition`

This sample was annotated purely manually, without using the *prodigy* annotation tool. Thereby, as in the above examples in the annotation rules, terms were simply enclosed with two square brackets, allowing to annotate nested terms as well. In addition, manual annotation was more convenient at this point, as it was also used to iteratively refine the annotation rules described above (see Chapter 2.4.1). In the ads sampled on the basis of ICT keywords, a large number of additional ICT terms could be found. Table 2 shows the number of found ICT terms in this first gold standard sample. To be used subsequently as input for the first NER model, the annotated text file was transformed into a JSONL format, following the annotation standards of *spaCy*. Since it does not allow nested terms, only the simple terms and the inner terms of nested terms were considered.

	Total Count	**Unique terms**
Simple terms[a]	1,115	831
Nested terms	51	49
Total	1,164	880

[a] *including inner terms of nested terms*

Table 2: Number of ICT Terms in Sample-0 (manually annotated gold standard)

3 Methodological Background

The main methods on which this work is based are briefly presented below. As this is a rather application-oriented thesis, it does not go into much detail, but the goal is to describe the high-level ideas behind the methods.

3.1 Text Embeddings

According to the distributional hypothesis of language (Firth [1957] and Harris [1954]), the meaning of a word can be inferred from the contexts in which it is used. Based on this property, embedding-based methods represent words as vectors such that similar words – words having a similar distribution – have similar vectors. Such embeddings play a key role in NLP, as they allow to convert textual data to meaningful numerical vectors, that can be processed by the computer. For a more formal application-oriented introduction to word embeddings, see Goldberg [2017] 65 ff.

Generally, two types of embeddings can be distinguished. *Static* word embeddings, generate the same embedding for the same tokens – regardless of their context. This type of embeddings, while relatively efficient computationally, has difficulty dealing with polysemous words, where meaning depends on context. To deal with this problem, *contextualized* embeddings have been developed, where the representation changes with the context (Wang et al. [2020]). Both types of embeddings were employed for different tasks in the present study and are described in more detail below.

3.1.1 Static Word Embeddings

Modern word embeddings are often learned by neural networks. One of the most popular approaches is *Word2Vec*, introduced by Mikolov et al. [2013]. The *Word2Vec* architecture allows to efficiently train word embeddings from a large corpus. The core idea is to train a neural network with one single hidden layer to perform a simple

task. Two models are used for this purpose: With *CBOW* the neural network learns to predict a word, given its context, whereas with *skipgram* it learns to predicts the context words given the central one. The weights of the neural network's hidden layer can the be used as the word embeddings (Mikolov et al. [2013], Wang et al. [2020]).

3.1.1.1 fastText Embeddings

fastText extends the *Word2Vec* architecture, by additionally representing each word as a bag of character n-grams. A vector representation is associated to each of these n-grams and the words are represented as the sum of them (Bojanowski et al. [2017]). Thus, through subword information, also the morphology of words is taken into account when computing the embeddings. In addition, this approach allows to compute word representations for words that did not appear in the training data. Especially for morphologically rich languages, which include German, this approach leads to better results (Bojanowski et al. [2017]).

In the context of ICT terms, *fastText* embeddings are expected to be advantageous, since these terms potentially contain a large number of slightly different spellings, rare compounds and combinations of various components that may only be partially known. As a results morphologically related terms, will be represented by similar embeddings, even if they have rarely occurring spellings. Therefore a list of known ICT term can be easily extended with new terms based on the similarity of their embeddings.

Case-sensitive *fastText* embeddings have been used for the expansion of the ICT term lists (see Chapter 3.3.1). These embeddings, which were trained on the entire German job ad corpus, were made available to me as part of the project.

3.1.2 Contextualized Embeddings

Contextual embeddings, extracted from pretrained language models have shown significant improvement of performance in various NLP tasks (Wang et al. [2020]). One of the most prominent language model is BERT (Bidirectional Encoder Representations from Transformers), introduced by by Devlin et al. [2018]. BERT embeddings have been used to train the NER model (see Chapter 3.2.2).

3.1.2.1 BERT Embeddings

BERT is designed to pretrain deep *bidirectional* representations from unlabeled text by jointly conditioning on both left and right context in all layers. To do this it uses a "masked language model". It randomly masks some of the subtokens from the input and the model needs to predict the original vocabulary id of the masked word based only on its context (Devlin et al. [2018]). The ability to capture information from both sides, distinguishes BERT from previous models, such as the GPT model (*Generative Pre-Training*) by Radford et al. [2018], that is based on a left-to-right transformer architecture or ELMo (*Embeddings from Language Models*) by Peters et al. [2018], which concatenates a left-to-right and a right-to-left model. With stronger capability of capturing information from both sides, BERT has beaten previous state-of-the-art on various NLP tasks (Devlin et al. [2018]).

For the present study the pretrained German BERT model *bert-base-german-cased*, provided by the HuggingFace transformer library (Wolf et al. [2020]), was used. *bert-base-german-cased* is based on German Wikipedia data (6 GB of raw txt files), the "OpenLegalDatadump" (2.4 GB) and news articles (3.6 GB)[1].

3.1.2.2 Domain-specific BERT Embeddings

As Gururangan et al. [2020] have shown, a second phase of domain-adaptive pretraining increases the performance of pretrained language models on tasks from the target domain under both high- and low-resource settings. In their study, they have continued pretraining RoBERTa[2] on four different domain-specific corpora. This domain adaptive pretraining could improve task performance in all domains, with the effect being particularly pronounced for the domains that differed the most from RoBERTa's source domain.

Within the framework of the NRP77 project the German BERT model *bert-base-german-cased* has been adapted to the domain of job ads by continued pretraining on a representative sample of the SJMM and OA job ad corpus. The resulting domain-specific JobadBERT has been used to fine-tune the NER model for recognizing ICT terms (see Chapter 3.2.2).

[1] for more information see `https://huggingface.co/bert-base-german-cased`

[2] RoBERTa is based on the same transformer-based architecture as BERT. It is was trained on over 160GB of uncompressed text, with sources ranging from English language encyclopedic and news articles, to literary works and web content (Liu et al. [2019]).

3.2 Named Entity Recognition

Named Entity Recognition (NER) plays a key role in information extraction, as it allows for the identification of "entities" in a text (Schmitt et al. [2019]). A named entity is anything that can be referred to with a proper name, like a person, a location, an organization. However, the term named entity is commonly extended to include things that are not entities per se (Jurafsky and Martin [2020], 153). Also the detection of ICT terms can be approached as a NER task. Like typical named entities, an ICT term often consists of multiple words. A NER model therefore must be able to find and label not only single words but *spans* of text. Furthermore it should be able to deal with word sense ambiguity (Jurafsky and Martin [2020], 153 ff.). In the context of ICT terms in job ads, this means that a particular word or phrase may or may not be classified as an ICT term, depending on its context. For example, the word *Applikation* can refer to an ICT program, but it can also refer to the application of medicines by a healthcare professional or to a painting method. Another example is the word *Netzwerk*, which can refer to a computer network, but can also mean a real network of people a person knows.

3.2.1 NER with Transition-based Parsing

Generally there are different approaches to tackle a NER problem, including rule-based or sequence labeling approaches. Today, neural architectures are becoming increasingly important in this context as well (Yu et al. [2020]). Lample et al. [2016] have introduced a NER-model that uses an algorithm inspired by transition-based parsing with states represented by Stack-LSTMs. Transition-based parsing is relatively less studied in NER (Honnibal [2017]), but builds also the basis for the spaCy NER model employed in this study (see next Chapter 3.2.2). Therefore, the idea behind transition-based parsing and its implementation for NER is described in more detail below.

Transition	Output	Stack	Buffer	Segment
	[]	[]	[Mark, Watney, visited, Mars]	
SHIFT	[]	[Mark]	[Watney, visited, Mars]	
SHIFT	[]	[Mark, Watney]	[visited, Mars]	
REDUCE(PER)	[(Mark Watney)-PER]	[]	[visited, Mars]	(Mark Watney)-PER
OUT	[(Mark Watney)-PER, visited]	[]	[Mars]	
SHIFT	[(Mark Watney)-PER, visited]	[Mars]	[]	
REDUCE(LOC)	[(Mark Watney)-PER, visited, (Mars)-LOC]	[]	[]	(Mars)-LOC

Figure 3: Example of a transition sequence for *Mark Watney visited Mars* with the Stack-LSTM model. (Source: Lample et al. [2016])

In a typical transition-based parsing process, input words are put into a queue and

partially built structures are organized by a stack. A set of actions are defined, which consume words from the queue and build the output parse (Zhang and Nivre [2011]). In the context of NER, this means, that instead of focusing on a single word, a chunking algorithm with different states is used to process the input sequence. To illustrate the procedure, Figure 3 shows an example of a transition sequence. The chunking algorithm makes use of a stack, an output-stack and a buffer that contains the words that have yet to be processed (Lample et al. [2016]). It starts in a condition whit no output and nothing on the stack, but all words of the sequence in the buffer. Then it proceeds by looking at the first word in the buffer. Now, a set of different transitions can be applied. In the case of NER, the following transitions are used: The SHIFT transition corresponds to the start of an entity and moves the word from the buffer to the stack. The OUT transition moves a word from the buffer directly into the output stack while the REDUCE(y) transition pops all items from the top of the stack creating a "chunk", labels it with label y, and pushes a representation of this chunk onto the output stack (Lample et al. [2016]). The algorithm completes when the stack and buffer are both empty. To predict this transitions, a neural network, which receives the embedded words of a sequence as input, is trained. Lample et al. [2016] have used a Stack-LSTM model (see Dyer et al. [2015]) for this purpose, whereas *spaCy* makes use of a simpler multi-layer perceptron (Honnibal [2017]). The goal of the training is to maximize the conditional probability of sequences of reference actions extracted from a labeled training corpus given the input sentences (Lample et al. [2016]).

3.2.2 spaCy NER Model

For training an ICT term recognition model, a *spaCy* transformer-based pipeline has been used. *spaCy*[3] is an open-source, Python based library for NLP. *spaCy v3.0* features different transformer-based pipelines, which allow to use transformer embeddings like BERT. They also interoperate with the HuggingFace transformers library, giving direct access to pretrained models, such as the *bert-base-german-cased* model, described in Chapter 3.1.2.1.

The pipeline used in the present study builds on the *spaCy* `transformer` and `ner` components. After tokenization with `spacy.Tokenizer.v1`, the `transformer` component first loads the pretrained transformer model (JobadBERT or bert-base-german-cased). Based on this language model, context-sensitive representations are computed for the training data. With this approach, knowledge gathered from large

[3]see `https://spacy.io`

amounts of raw text is available in the pipeline, so that the model is able to generalize better from the few annotated examples. The downstream `ner` component can then use these representations as input features to improve its predictions. The used NER model is based on is based on the `spacy.TransitionBasedParser.v2`. It builds on a transition-based parser model, as described above (Chapter 3.2.1).

spaCy v3.0 allows to define all settings and hyperparameters for the training in one configuration file, which can be passed directly to `spacy train` on the command line. The settings and hyperparameters used in the present study are mainly based on the default settings and can be found in the the configuration file in the appendix B. If not stated otherwise, all the NER models have been trained using these settings.

3.3 Prodigy Recipes

Prodigy is an annotation tool for creating training and evaluation data for machine learning models. It has been developed by *Explosion*, who is also behind *spaCy*. Its self-declared goal to be *"so efficient that data scientists can do the annotation themselves, enabling a new level of rapid iteration"*. Unlike *spaCy*, *prodigy* is not free or open-source, although some parts of the source code are available[4]. *Prodigy* provides a range of built-in recipes, which are Python functions that can be run via the command line. In the context of this thesis, I used two of them, which are described in more detail below.

3.3.1 Prodigy terms.teach

The *prodigy* `terms.teach` recipe[5] allows to interactively build or extend a terminology list. It uses word vectors and already known terms as seed terms. Based on the seed terms, a target vector is created and only terms similar to that target vector are proposed to the annotater. These terms can then either be accepted or rejected. For the annotation process *prodigy* provides an easy-to-use graphical annotation interface (see Figure 4). When annotating, the recipe iterates over the vector model's vocab and updates the target vector with the accepted words. The results are stored in a SQLite-database and can be easily accessed or downloaded as a JSONL-file after annotating.

[4]For this thesis, I benefited from the fact that the Department of Computational Linguistics has an according license, allowing me to work with *prodigy*

[5]see `https://prodi.gy/docs/recipes#terms-teach`

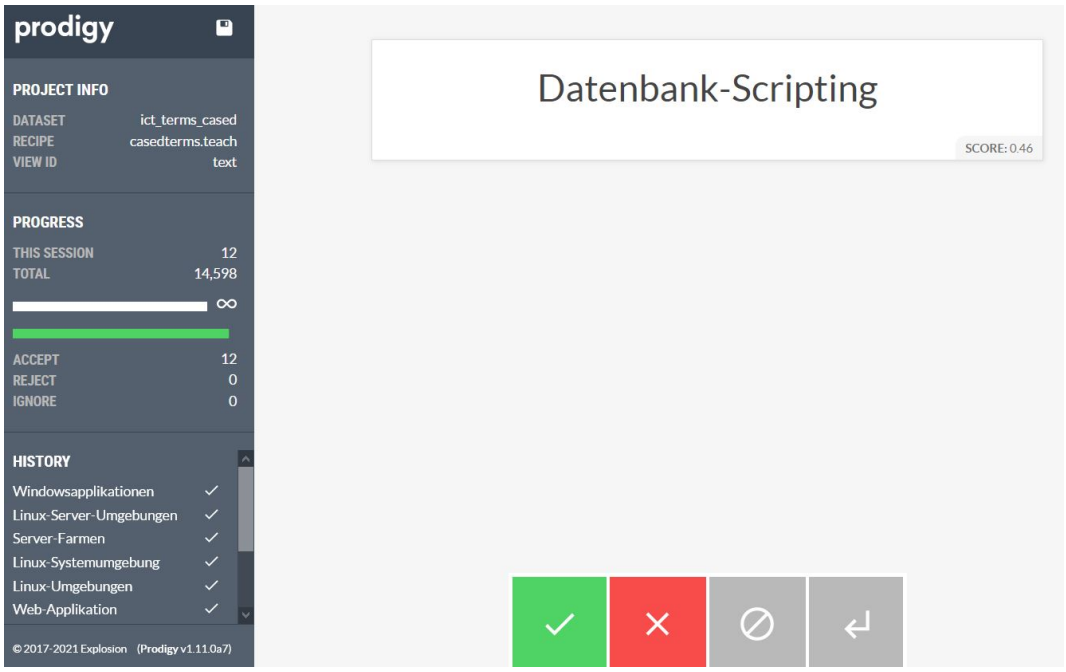

Figure 4: Prodigy terms.teach Annotation Interface

Per default `terms.teach` is not case-sensitive. However, for many ICT terms upper and lower case may be important to distinguish them from other terms (e.g. the capitalized Letters "R" and "C" can stand for programming languages). Furthermore, the default method only considers terms that exclusively consist of alphabetic characters, however many ICT terms contain numbers and hyphens (e.g. "3D-CAD", "TYPO3", "E-Learning"). Therefore the `terms.teach` Python script has been adapted to add case-sensitivity and to accept alphanumeric characters and hyphens as part of a term.

To be used as input for `terms.teach` a *spaCy* model had to be generated from the *fastText* job ad word embeddings (see Chapter 3.1.1.1). The term lists, described in 2.3 and the terms found in the first manual gold standard (see Chapter 2.4) served as basis for the original seed term list. The procedure and the experience with the expansion of the term list will be described in more detail in Chapter 4.1.

3.3.2 Prodigy ner.correct

The *prodigy* `ner.correct` recipe[6], provides an annotation interface to manually correct entities predicted by a *spaCy* model. As input, *prodigy* expects data in JSONL format as described in Chapter 2.2.2. The suggestions of the model on the data can be removed and corrected if necessary. The corrections are stored in a SQLite-database and can later on be used as new gold standard data.

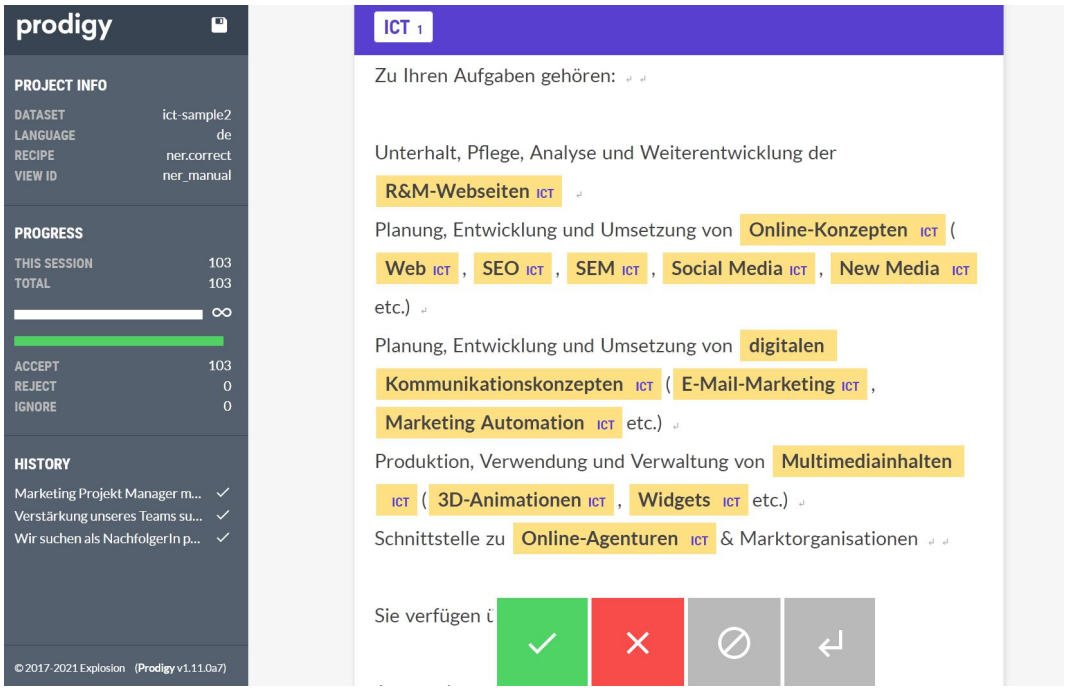

Figure 5: Prodigy ner.correct Annotation Interface

Using pre-annotations can cause potential bias by directing the annotation in a specific direction (Fort and Sagot [2010]). However, various studies have shown, that automatic pre-annotation in general helps to increase both speed and accuracy of annotations in a NLP context (see e.g. Rosset et al. [2013], Lingren et al. [2014], Fort and Sagot [2010]).

3.4 Mallet Topic Model

In order to specifically sample ads with a certain content focus (e.g. ICT ads), a topic model computed on the entire job ad corpus has been used. This model was made available to me as part of the NRP77 project. The idea and methodology behind this topic model is briefly described below.

[6]see `https://prodi.gy/docs/recipes#ner-correct`

A topic model looks for patterns in the use of words in a corpus and attempts to identify the thematic mix of each document. Hence, from the model's perspective a topic is a list of words that occur in statistically meaningful ways (Graham et al. [2012]). Probabilistic topic models are based upon the idea that documents are mixtures of topics, where a topic is a probability distribution over words Steyvers and Griffiths [2007]).

The utilized job ad topic model has been computed using the MALLET topic model package. MALLET is based on LDA (Latent Dirichlet allocation), a generative generative probabilistic model, developed by Blei et al. [2003]. Simply put, LDA is based on the assumption that each topic consists of a probability distribution over the corpus vocabulary, and each document contains a mixture of different topics. Therefore, based on the occurrence of words in a document, its topics can be inferred, since it is assumed that each word in a document comes from a certain topic (Tomar [2018]). To assign words to topic, respectively topics to document, MALLET uses on an implementation of Gibbs sampling, a statistical technique to construct a sample distribution. Gibbs sampling is an algorithm for successively sampling conditional distributions of variables, whose distribution over states converges to the true distribution in the long run (Tomar [2018], Graham et al. [2012]).

When creating the topic model with MALLET, first, the German job ads corpus was converted into a binary mallet corpus file. Thereby not only general german stopwords, but also domain-specific stopwords have been removed. Then a MALLET topic model with 100 topics has been calculated. The result of this model is both a list of topics and the keywords composing those topics and a file that indicates for each document (in our case, job ads) what percentage of topics it consists of. By analyzing the keywords of the 100 topics, 4 highly ICT-specific topics (Chapter 4.2.4) and 10 other ICT-related topics (Chapter 4.2.6) were determined. Hence, the topic assignment served as the basis for drawing more targeted samples from the entire corpus.

4 Experiments and Results

4.1 Term List Expansion

4.1.1 Extension of the ICT Term List

As a starting point, the existing ICT term lists (see Chapter 2.3) were extended using prodigy's *terms.teach* recipe (see Chapter 3.3.1). As *terms.teach* builds up on similarities between word vectors, it can only handle single word terms, respectively compounds with a hyphen. Therefore only the single-word terms from both ICT term lists, plus all single terms found in the first gold standard (Sample-0) have been used as initial seed terms. Within two hours of annotation, based on these 2,516 seed terms, 4,220 new ICT terms could be found. Table 3 gives an overview of the processed terms. As one can see, the suggestion from the model based on vector similarity were quite good, in total 93% of all suggested terms could be accepted. Many of these terms were long, rather specialized terms, composites or terms with unusual spellings or misspellings (e.g. *Dokumentenverwaltungssoftware, Workflow-Management-Applikationen, Anivirus-Serverlandschaft*). Only 2% of the suggested terms were clearly wrong and had been rejected. The remaining 5% of terms were mainly borderline cases which were ignored in case of doubt, in order not to interrupt the annotation process for too long.

Before this expanded ICT term list could be used for the next steps it was first merged with the original ICT term lists containing also multi-word terms. In addition, a simple automatic extension was made by adding an additional spelling for all terms containing a hyphen, where the hyphen was replaced by a space (e.g. "Windows-Programmierung" leads to the additional term "Windows Programmierung"). On the other hand, all the terms that were present in the gold standard were excluded at this point, in order not to bias the results, since the gold standard data would serve as training data in the next step. As a result the final ICT term list contained a total of 10,171 terms.

ICT seed terms	**2,516**	
Processed terms	**4,519**	100.0%
Accepted	4,220	93.4%
Rejected	83	1.8%
Ignored	216	4.8%
Seed terms + accepted terms	**6,736**	
Add multi-word terms from original list	1,529	
Add terms through automated expansion	2,762	
Exclude terms in gold standard (Sample-0)	-856	
Final ICT term list	**10,171**	

Table 3: Expansion of ICT terms with prodigy terms.teach (2 hours)

4.1.2 Creation of a non-ICT Term List

The ICT term list described above has been used as additional training material for a first NER model (model At, see Chapter 4.2.3). In order to give the model something to learn from the content of these terms instead of just learning that a simple term (instead of a full ad) equals ICT term, additionally a *negative term list* has been created. This list should include examples of non-ICT terms to counterbalance the long list of ICT terms.

In a first attempt, the 83 rejected terms of the expansion described above have been used as seed terms to find new non-ICT terms with the *terms.teach recipe*. However, this approach proved to be unsuitable because the newly proposed terms were almost exclusively ICT terms. Since all of the seed terms were originally proposed as ICT terms, their vectors per definition are very similar to real ICT terms, making it hard to find new non-ICT terms. Therefore, this annotation attempt was abandoned. Instead, all nouns (recognized by *spaCy* POS-tags 'NN', 'NE', 'NNE') of the gold standard that were not categorized as ICT terms were used as a seed terms. These 1,810 seed terms could be extended to a total of 4,651 non-ICT terms within one hour using *terms.teach*. Similar to the expansion of the ICT term list, 93% of the proposed terms could be accepted (see Table 4). With the additional automated expansion (add version of composite with blank instead of hyphen), the final non-ICT term list consisted of 5,061 terms.

Non-ICT seed terms	**1,810**	
Processed terms	**3,036**	100.0%
Accepted	2,841	93.6%
Rejected	80	2.6%
Ignored	115	3.8%
Seed terms + accepted terms	**4,651**	
Add terms through automated expansion	410	
Final Non-ICT term list	**5,061**	

Table 4: Expansion of Non-ICT terms with prodigy terms.teach (1 hour)

4.2 NER-models with domain-specific BERT Embeddings

The core idea behind my approach is to train a NER model (see Chapter 3.2.2) and iteratively improve its performance by adding more training data through correcting the models' predictions on new data samples with `prodigy ner.correct` (see Chapter 3.3.2). These rounds of corrections have also been used to qualitatively evaluate the output of the models. The iterative process of training and increasing the training data is illustrated in Figure 6.

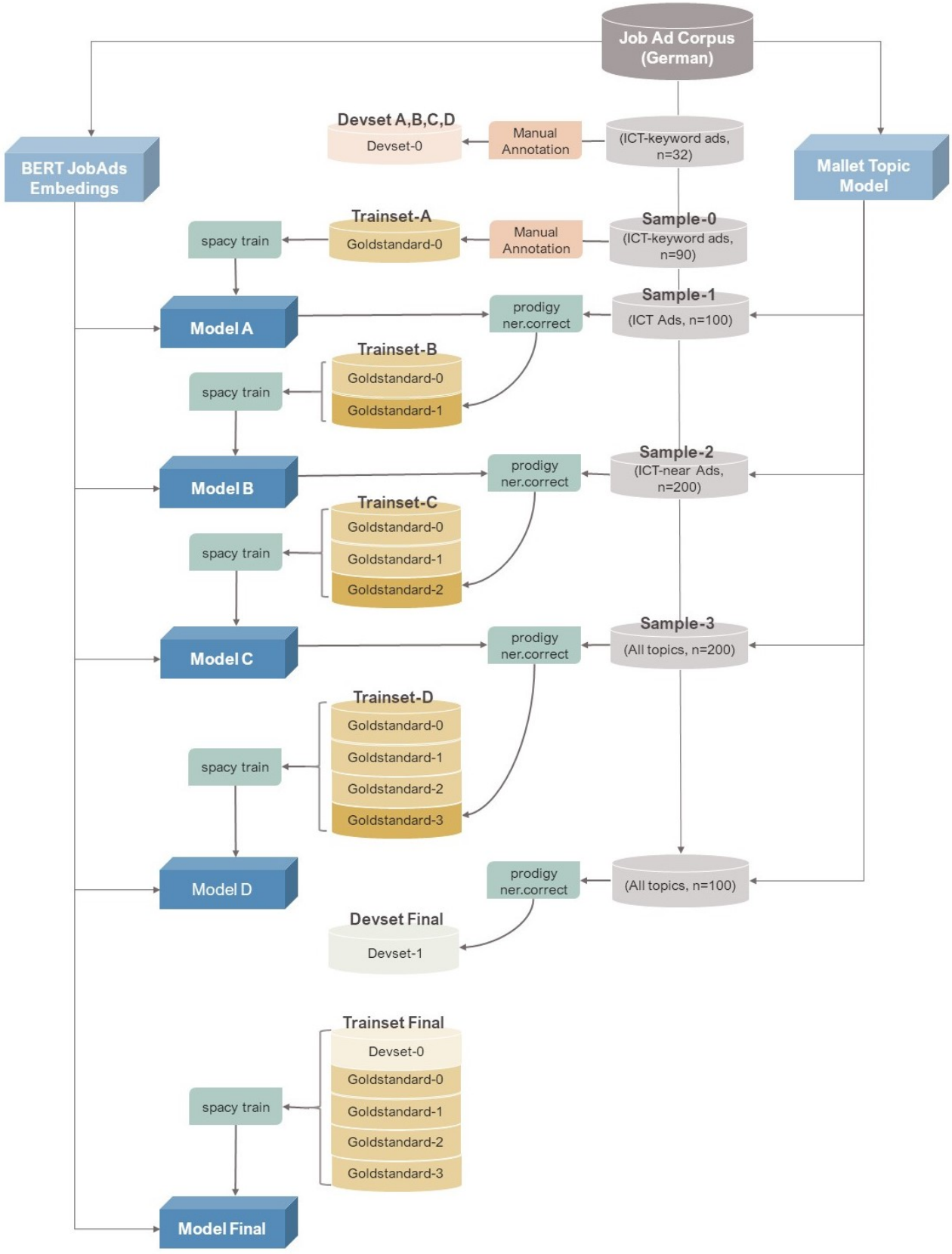

Figure 6: Iterative Training and Annotation Process

4.2.1 Overview Models

The different models that have been created throughout the process, are all named according to the same principle. The first, capital letter, e.g. **A** denotes the training sample it is based on, and the additions **t** and **n** denote whether the **t**ermlist, respectively the **n**on-ICT term list were added to the training data. Model **Atn** for instance refers to the model trained on Sample-0 and both term lists. Table 5 gives an overview of the models names and the training data they are based on.

First letter	Training data	n	Devset
A	Sample-0 (Manual Gold Standard)	90	Devset-0
B	Sample-0 + Sample-1 (ICT ads)	190	
C	Sample-0 + Sample-1 + Sample-2 (ICT-near ads)	390	
D	Sample-0 + Sample-1 + Sample-2 + Sample-3 (All topics)	590	
F	Sample-0 + Sample-1 + Sample-2 + Sample-3 + Devset-0	622	Devset-1
Additions			
t	ICT term list	10,171	
n	Non-ICT term list	5,061	

Table 5: Overview Model Names and Training Data

4.2.2 Used Evaluation

All models (A-D), except the final model (F) have been evaluated on the same development set, referred to as Devset-0. It consists of 32 annotated ads, sampled according to the same principle as the manual gold standard (Sample-0), described in Chapter 2.4. For the final model a new development set (Devset-1) has been created, it consists of 100 ads sampled across all topics, as described for Sample-3 (see Chapter 4.2.8).

The models are each evaluated by recall, precision and F-score. Recall is the ratio of correctly labeled terms to all terms that should have been labeled, precision is the ratio of the number of correctly labeled terms to the total labeled, and the F-score is the harmonic mean of the two (Jurafsky and Martin [2020]).

Since the training network is initialized randomly, the result is not deterministic, leading to variance due to random chance. Therefore every model was trained 3 (models A-D) or 5 (model F) times, to minimize the influence of differences caused by random initialization. The reported precision, recall and F-score are the mean values of the models, additionally the F-Score standard deviation (σ) is indicated.

4.2.3 Results of Models A

The first, baseline model A was trained on the 90, manually annotated Sample-0, sampled by using ICT keywords. It reached an average F-score of 84.36. The inclusion of the ICT term slightly improved the overall performance of the model (from 84.36 to 84.96 with the ICT term list and to 85.13 with the additional non-ICT term list). Additionally a recall oriented model was computed by changing the default training.score_weights from determining the overall score entirely by the F-score (`ents_f=1.0, ents_p=0.0, ents_r=0.0`) to counting the recall with 25% (`ents_f=0.75, ents_r=0.25`) towards the overall score. When computing the first Atn_recall model, this significantly raised the recall to 88.37, however that turned out to be more of a coincidence, as this effect could not be observed when training the model with the same settings two more times. Hence, the average recall of model Atn_recall as shown in Table 6 does not stand out anymore.

Model	Precision	Recall	F-score	σ F-Score
A	85.77	83.00	84.36	0.34
At	86.27	83.75	84.97	0.59
Atn	85.79	84.49	85.13	1.02
Atn_recall	82.85	85.81	84.27	0.47

Table 6: Results of Models A

4.2.4 First Expansion of Training Data with ICT Ads

In order to expand the training set, a first sample with 100 ICT ads, based on the topic model described in Chapter 3.4, has been selected. For this purpose among the 100 topics, 4 topics with a strong ICT component have been chosen based on their vocabulary (see Table 7). For every ICT topic, 25 ads with at least a share of 40% of the topic, were randomly sampled in the whole corpus (ads selected for Sample-0 were excluded). Independently of this, samples have been selected by considering all different sources (SJMM, SJMM-O, OA) and years. For the exact distribution of sampled ads among sources and years, see Table 15 in the appendix.

Then, this sample was pre-annotated[1] with model A (50 ads) and model Atn_recall (50 ads) and corrected with `prodigy ner.correct` (see Chapter 3.3.2). The entire

[1]While the indicated scores in the result sections refer to the average scores over several models, for the pre-annotation, only the first models were used in each case.

Topic	Top 25 words
Topic 4	support gute kunden kenntnisse aufgaben level bereich informatik erfahrung vorteil windows software service deutsch abgeschlossene ausbildung informatiker unterstützung installation microsoft server suchen ict office betreuung
Topic 21	kenntnisse erfahrung gute system betrieb security engineer bereich server kunden microsoft aufgaben windows informatik management deutsch profil services linux level arbeitsort lösungen netzwerk vorteil anforderungen
Topic 24	erfahrung software kenntnisse gute entwicklung java kunden informatik net aufgaben technologien design anforderungen deutsch bereich sql vorteil javascript engineer lösungen applikationen team profil agilen entwickler
Topic 25	sap kunden erfahrung gute kenntnisse bereich erp aufgaben anforderungen berater informatik abap umfeld crm projekten profil vorteil bewerben bewerbung consultant lösungen suchen business support customizing

Table 7: ICT topics from Mallet topic model

correction process took about 2 hours. It was expected, that the high recall score of the second model would potentially facilitate the annotation process, as it is easier to correct wrongly annotated terms than finding missing terms in the ads. Yet, my subjective feeling during annotation could not confirm this, I found both pre-annotations to be of approximately similar quality. A quantitative evaluation of the two models' pre-annotations over all corrected annotations, however showed that the performance of model Atn_recall was slightly better with reaching an F-Score of 86.68, compared to 87.66 from model A. Hence, for both models, the evaluation on the corrected sample shows a better performance than on the development set. This may be due to the nature of the ads, but to a certain extent an annotator effect may also plays a role. Suggestions of a model are more likely to be accepted in borderline cases and as a consequence, the result may differ slightly from a annotations without pre-annotation. Some observations made during the correction process with `prodigy ner.correct` are briefly described below. As expected, most of the ads in this sample were strongly ICT-oriented, with many ads from the ICT industry. The pre-annotation of both models was in general already quite good. Figure 7 shows a perfect annotation of the model for an ad with very specific ICT terms. The model also seems to have already learned how to deal with ellipses according to the annotation guidelines, as the correct annotation of "Datenextraktion, -modelling" in 8 shows. However, just below this example, a segmentation error can be seen, where the ICT term SAP S/4HANA is not correctly delimited from the next token.

Figure 7: Annotations of Model A on Sample-1 – Example 1

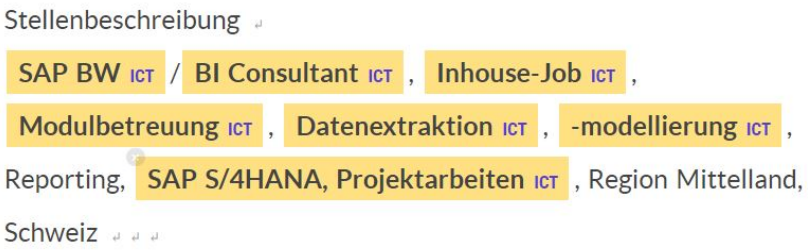

Figure 8: Annotations of Model A on Sample-1 – Example 2

4.2.5 Results of Models B

Model B has been trained on Sample-0 and the newly generated Sample-1 (ICT ads). Compared to model A, there is a considerable improvement. The F-score could be increased from 84.36 to 86.36, which corresponds to an increase of 2 percentage points. The additional inclusion of the ICT term list, however could not improve the model substantially. Table 8 shows the detailed results of the different models B.

Model	Precision	Recall	F-score	σ F-Score
B	87.62	85.15	86.36	0.37
Btn	86.72	84.49	85.58	0.13

Table 8: Results of Models B

4.2.6 Second Expansion of Training Data with ICT-near Ads

With the goal to further expand the training set, a new sample (Sample-2) with 200 ”ICT-near” ads based on the topic model described in Chapter 3.4 has been selected.

For this purpose among the 100 topics, 10 new topics with some ICT aspects in it – but with a weaker ICT focus than for Sample-1 have been selected. The vocabulary of these topics can be found in Table 18 in the appendix.

For each of these 10 topics, 20 ads with a share of at least 40% of the topic were randomly selected from the entire corpus, excluding ads from Samples-0 and 1. Independently of this, again a coverage of all different sources (SJMM, SJMM-O, OA) and years was respected. For the exact distribution of sampled ads among sources and years, see Table 16 in the appendix.

The new sample has been pre-annotated with model B and then, corrected with `prodigy ner.correct` (see 3.3.2) within 3 hours. An evaluation of the pre-annotations of this model showed an F-Score of 79.38, which is clearly worse than for the first sample. However, this can be explained by the fact that model has been trained mainly on ICT ads so far and has now been confronted with new ad types and ICT terms.

About a quarter of the ads in this sample (56 out of 200) did not contain any ICT term. Moreover, compared to the ICT sample (Sample-1), many ads use comparatively few and rather general ICT terms (e.g. *MS Office-Paket, IT-Flair, PC-Kenntnisse*) or branch-specific ICT skills and tools (e.g. *ERP-Software, 3D CAD, Schmaschinenoptimierung*) appear. In addition, new ICT aspects are showing up, such as *E-Paper, digitale Medien, digitale Transformation, Onlinebereich, Social Media, Digitalisierung, digitale Bankprodukte, E-commerce*. Although these kind of terms are often recognized correctly by the model too, there tends to be more errors than with the ICT terms, which were very present in the previous samples. For instance in the example shown in Figure 9, the model has incorrectly split the term "social media".

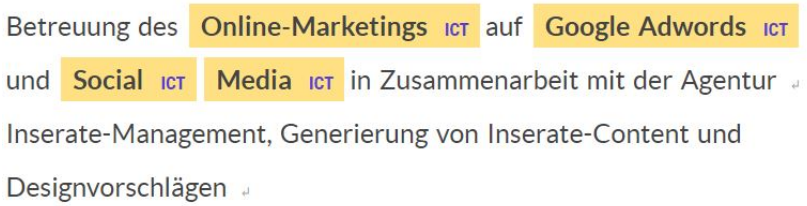

Figure 9: Annotations of Model B on Sample-2 – Example 1

The incorrect annotation of the term "Growth Mindset", as shown in Figure 10 illustrates a common type of error: The model often tends to incorrectly annotate English expressions as ICT term. Due to the large number of English ICT terms in the German corpus, this is not surprising.

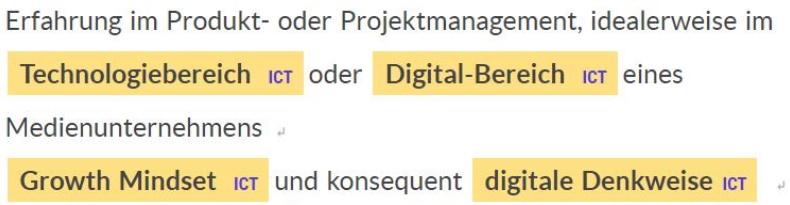

Figure 10: Annotations of Model B on Sample-2 – Example 2

4.2.7 Results of Models C

Model C, which was trained on Sample-0 and 1 plus the the new sample with ICT-near ads (Sample-2), reached a performance of 85.31. The inclusion of term lists in the Ctn model did not substantially improve the model. Overall, both models perform slightly worse than the Models B and Btn. This drop in performance can probably be explained by the increasing discrepancy between training and development data (more on this in the discussion of the results in Chapter 5.2).

Model	Precision	Recall	F-score	σ F-Score
C	87.95	82.92	85.31	0.03
Ctn	86.95	84.57	85.73	0.61

Table 9: Results of Models C

4.2.8 Third Expansion of Training Data with Ads from all Topics

The previous sampling was clearly ICT-recall oriented. The next step is to ensure that all semantic topics in the corpus are covered and any over-analyses are detected and remedied. Therefore, in the last extension of the training data, 200 ads (Sample-3) were selected among all 100 topics of the topic model (see Chapter 3.4). For every topic, two ads were sampled from the entire job ad corpus, excluding the ads of the samples 0, 1 and 2. As before, additionally, a coverage of all different sources (SJMM, SJMM-O, OA) and years was considered (see Table 16 in the appendix).

The new sample has been pre-annotated with model C and corrected with `prodigy ner.correct` (see Chapter 3.3.2). The correction of this sample took about 2 hours. Thus, compared to the previous samples, less time was required for correction, which could be due to my increasing experience in this, but certainly also due to the

relatively large number of ads without any ICT term. In contrast to the previous annotation rounds, this new sample contained, as expected, a significantly higher proportion of ads that were not ICT-related at all. More than half, or 115 of the 200 ads did not contain any ICT term. The remaining ads, with the exception of some ICT specific ads, contained rather general terms, such as *Internet, MS Office* or *Computerkenntnisse*, as illustrated by the example in Figure 11. These kind of terms were generally well recognized by the model. But the model was in general also able to handle ICT-rich ads very well, as shown in Figure 12. Errors occurred mainly in borderline cases or in term segmentation. Overall, Model C achieved an F-score of 85.06 when evaluating its annotations with against the corrected annotations of Sample-3. The inclusion of the ICT-near training data (Sample-2) for Model C, which already contained a certain proportion of ads without ICT terms, seems to have already helped the model in dealing with less ICT-rich ads.

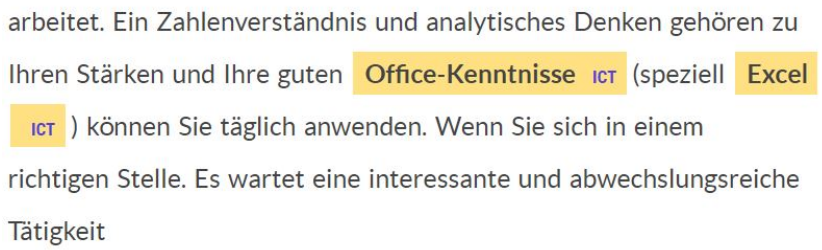

Figure 11: Annotations of Model C on Sample-3 – Example 1

Figure 12: Annotations of Model C on Sample-3 – Example 2

4.2.9 Results of Models D

Model D was trained on Sample-0, 1 and 2 and the new sample covering all topics (Sample-3). With an F-score of 86.79 it is slightly better than model B (F-Score: 86.36) and therefore reached the best performance of all models so far. The inclusion of term lists could not help to further improve the result.

Model	Precision	Recall	F-score	σ F-Score
D	89.42	84.32	86.79	0.33
Dtn	89.05	83.75	86.31	0.21

Table 10: Results of Models D

4.2.10 Results of Final Models

For the creation of a final model, a new development set was created. Its goal is to ensure that the final model can handle the whole range of ads as well as possible. It contains a total of 100 ads from all topics, and was extracted like Sample-3 (see 4.2.8). Also here, a coverage of all different sources (SJMM, SJMM-O, OA) and years was considered (see Table 15 in the appendix). To compute the final model, the 32 ads of the original development set were added to the training set, leading to a total of 621 annotated ads.

This final model without term lists (F) reached an average F-score of 90.42. The inclusion of the term lists in the model (Ftn) has not led to an improvement on average. However, the best individual performance was achieved by a Ftn model with an F-score of 91.73. At the same time, the inclusion of the term lists also led to the worst individual result (F-score of 88.82). The relatively high variance of the results of the Ftn models is also reflected by the large standard deviation of 1.01.

Model	Precision	Recall	F-score	σ F-Score
F	91.14	89.72	90.42	0.26
Ftn	90.99	89.66	90.31	1.01

Table 11: Results of Final Models

4.3 NER-models with Generic BERT Embeddings

To find out what influence the use of the domain-specific JobadBERT had, both the first model A and the final models F and Ftn were computed with generic BERT embeddings (BERT-base-german-cased, as described in Chapter 3.1.2.1).

Table 12 shows the results in comparison to the models trained on domain-specific BERT embeddings. Model A with generic BERT embeddings (A_BERT-base) reaches an F-score of 80.73, this is 3.63 percentage points less than the result of the same model with domain-specific BERT embeddings (A_JobadBERT). For the final model F, the difference is less, but still clearly visible. While the model with generic embeddings (F_BERT-base) achieves an F-score of 89.03, the result of the model with domain-specific embeddings (F_JobadBERT) is 1.38 percentage points better, reaching an F-score of 90.42. It is also interesting to see that the inclusion of term lists in the final model with generic BERT embeddings has a clearly negative impact on performance. Whereas in the model with domain-specific BERT the term lists have almost no impact on the result.

Model	**Precision**	**Recall**	**F-score**	σ F-Score
A_BERT-base	81.64	79.87	80.73	0.33
A_JobadBERT	85.77	83.00	84.36	0.34
F_BERT-base	88.92	89.16	89.03	0.41
F_JobadBERT	91.14	89.72	90.42	0.26
Ftn_BERT-base	88.94	87.08	87.98	1.10
Ftn_JobadBERT	90.99	89.66	90.31	1.01

Table 12: Results – BERT-base vs. JobadBERT

4.4 Ablation Study: ”Term-lists-only” Training Data

In order to better understand the influence of term lists on term recognition, an additional NER model was trained, using only the ICT and Non-ICT term lists as training data.

Evaluated on the first development set (Devset-0), which contains 32 ads, the model only reaches an F-score of 34.61. The same model has been evaluated against the final development set (Devset-1). Here, the result is even slightly worse. A further analysis showed, that 80 of the total 290 ICT terms in Devset-1 are also present in the term list, this is a share of 27.6% which corresponds quite closely to the recall

Model	Precision	Recall	F-score
tn (Devset-0)	41.95	29.46	34.61
tn (Devset-1)	32.19	28.92	30.47

Table 13: Result of model only trained on term lists

score. This could lead to the conclusion that the model simply memorizes the terms from the list and annotates them directly. However, only 11% of the 292 terms annotated by the model tn can actually be found in the term list.

An additional qualitative error analysis of the model's annotation showed that the model predictions are not completely wrong, but tend to include too much surrounding text to the ICT terms. Almost all annotated terms contain an ICT aspect, but the model struggles to set the boundaries correctly. Figure 13 shows how the first three terms are annotated correctly, but the fourth term then combines way too much text into one term. This is not an isolated case, but many ads contain such long terms, in extreme cases, as illustrated by the example in Figure 14, these terms extend over entire paragraphs. Thus, without context, the model does not seem to be able to learn the correct separation of ICT terms from their surrounding tokens.

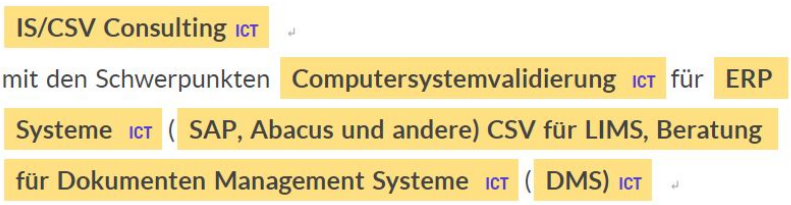

Figure 13: Annotations of Model tn – Example 1

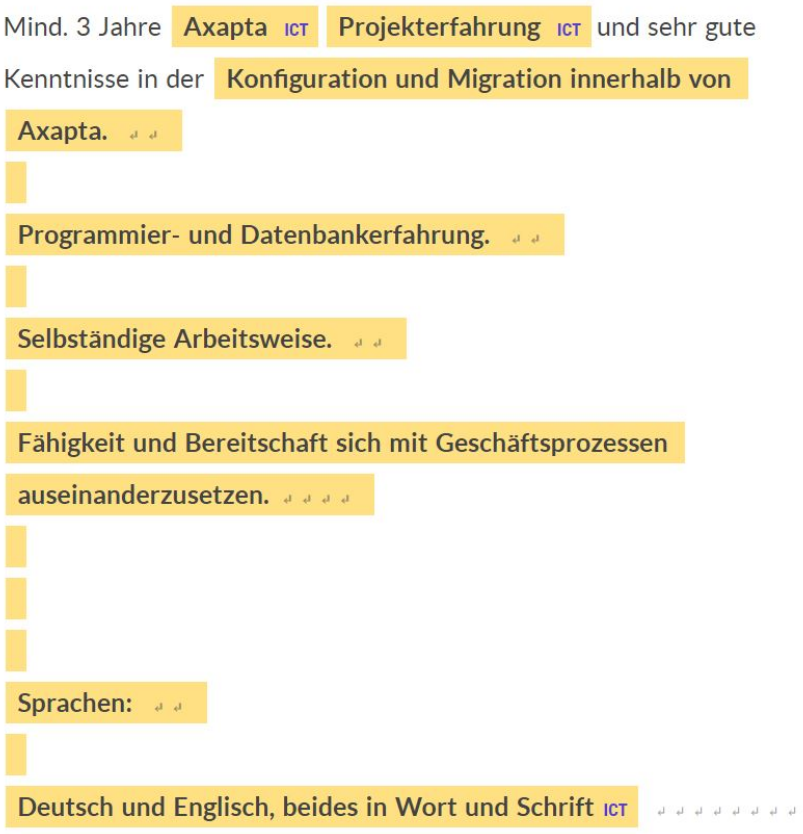

Figure 14: Annotations of Model tn – Example 2

5 Discussion

5.1 Sampling and Annotation of Training Data

Before interpreting the different results of the NER model, there is a brief discussion of the sampling and annotation strategy. The first sample of ads was drawn based on ICT keywords and was annotated manually. This approach proved to be useful; within the 90 ads, more than 800 different ICT terms could be found. These examples of terms helped to identify different problematic cases and to establish and refine annotation rules. The later samples were drawn based on the topic model, which allowed to sample new training material in a targeted way and to slowly shift the focus from very ICT-rich ads to a broad coverage across all topics. Within three rounds and a total of 7 hours of correcting the predictions of previously trained NER models, the training corpus could be increased from 90 to 590 ads. The use of *prodigy* has proven to be very practical and effective for this process. The only drawback is that the tool does not allow annotation of nested terms. In addition, the annotation of long ads was sometimes quite tedious, since a large number of potential ICT terms were to be seen and annotated within one view. To prevent this, it might be useful to divide the sampled ads into different segments before starting the annotation process.

The annotation rules were very helpful for the further annotation process, however the annotation of ambiguous ICT terms remained challenging, since it is very hard to formulate general rules. Additionally, the tooling used in the next steps has limited the conceptualization of ICT terms in that they do not allow for nested terms. Nevertheless, the approach of conceptualizing the recognition of ICT terms as a NER tagging problem has proven to be useful overall.

5.2 Results of NER Model

In the following section the results of the different NER models, described in Chapter 4 are compared and discussed.

5.2.1 Models based on domain-specific BERT

Figure 15 shows the development of the performances from models A to D, trained with domain-specific BERT Embeddings. There is a clear improvement from model A (trained on 90 ICT-keyword based ads) to model B, where additionally 100 ICT ads were added to the training corpus. Not only the F-score, but also precision and recall improve evenly by around 2 percentage points each. From model B to model C, however, there is a drop in the F-score, which is mainly caused by a clearly lower recall. By contrast, precision continues to rise slightly. This phenomenon is most probably related to the nature of the development set on which the evaluation is based. While the development set was sampled based on ICT-keywords, the new training data used in model C, contained not only fewer, but also some new types of ICT terms. This results in the new model being less able to recall the terms in the very ICT-rich development set. For Model D, where 200 ads from all topics have been added to the training corpus, the trend goes up again. With a slightly better F-score than Model B, it has the best performance of all these four models. Hence, the overall increase in the size of the training material appears to compensate for the different nature of the training and development set, which is still an issue.

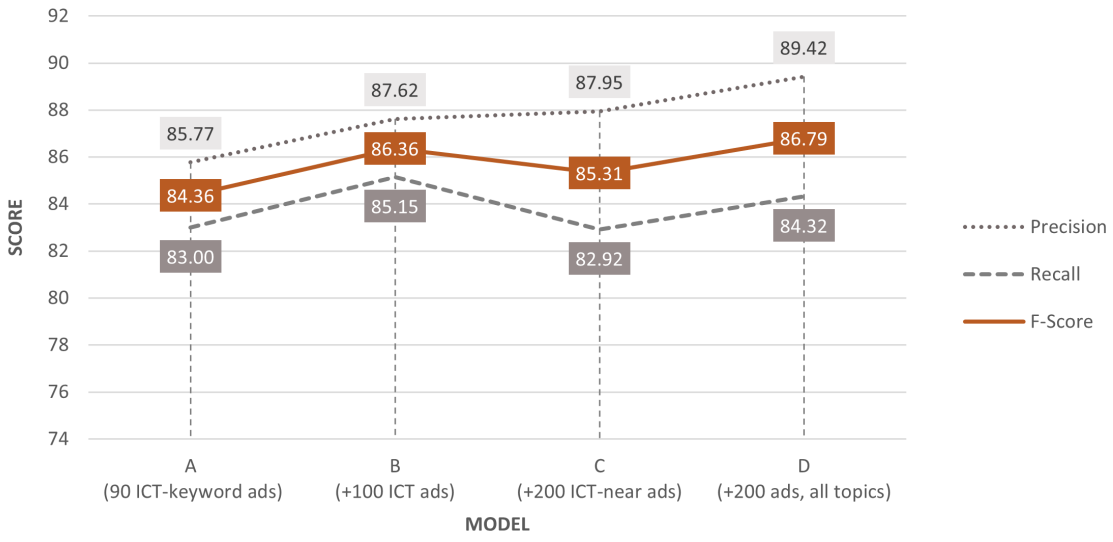

Figure 15: Performance of JobadBERT Models A to D, evaluated on Devset-0

Finally, for the final model, a new development set (Devset-1), with ads from all topics has been created. This is to ensure that the final model can handle the full range of ads as well as possible. To examine how the previous models perform on this final development set, the predictions of models A to D were also evaluated against this development set. Thereby, of course, it must be kept in mind that

the best models A to D were selected based on their performance on the original development set. Figure 16 shows that a clear learning curve from model A to the final model F is visible now. Different than with the evaluation on the first development set (Devset-0), shown in Figure 15, the precision score starts at a much lower point than the recall, with both values increasingly converging. This effect is probably related to the fact that the training material is becoming not only larger but increasingly diverse in terms of topics. For the final model F, precision is again slightly above recall. Overall the final model achieved an F-score of 90.42. By increasing the size of the training corpus from 90 to 621 ads, the 90% limit could thus just be exceeded.

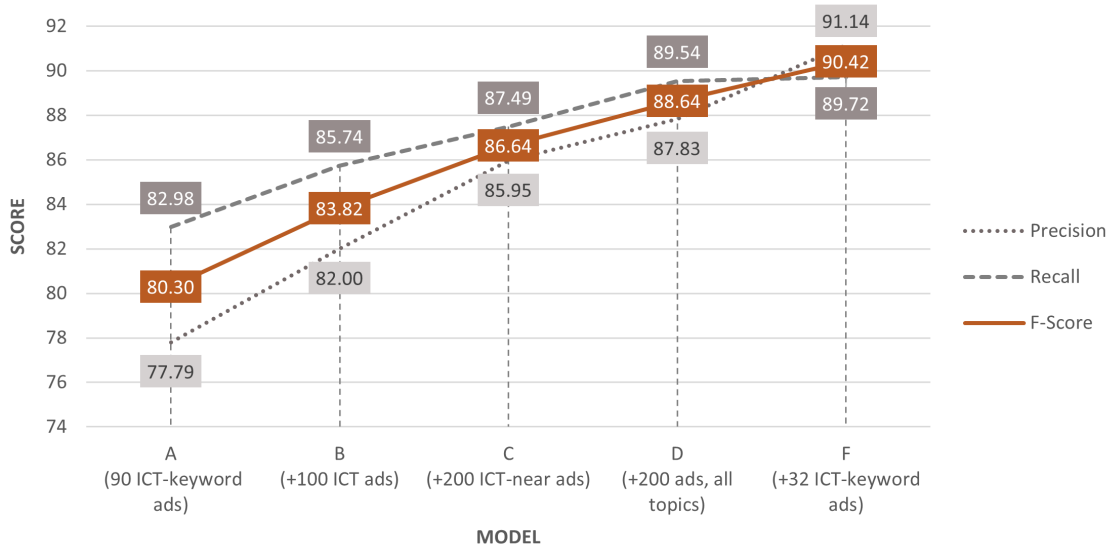

Figure 16: Performance of JobadBERT Models A to F, evaluated on Devset-1

5.2.2 Generic vs. domain-specific BERT

The use of domain-specific BERT compared to generic BERT embeddings results in a significant improvement of the models' performance. Figure 17 shows the results of the first model A and the final model F for both language models.

Model A, trained on only 90 ads, using BERT embeddings achieves an F-score of 84.36, while the same model computed with generic BERT is a 3.63 percentage points below this score. By increasing the size of the training material for model F, the difference between the two models decreases, however the model with the domain-specific JobadBERT still outperforms the model with generic BERT by 1.39 percentage points. This demonstrates the power of domain-specific BERT embeddings in low-resource settings.

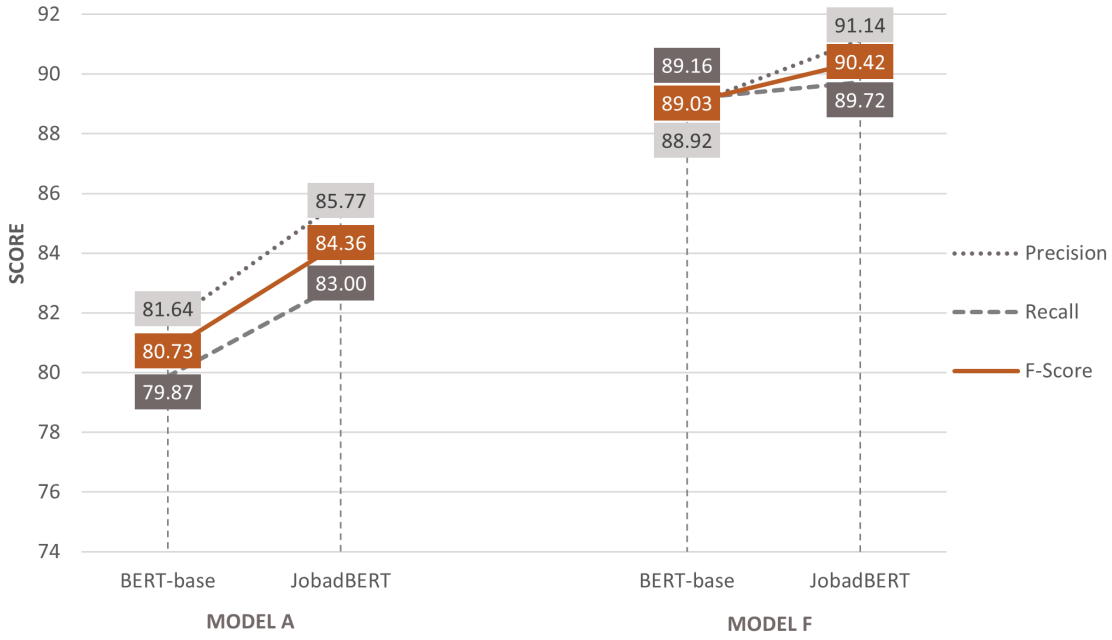

Figure 17: Performance of generic BERT (BERT-base) vs. domain-specific BERT Models (JobadBERT) for models A and F

5.2.3 Benefit of Term Lists

In contrast to the BERT embeddings, the inclusion of term lists in the training corpus did not provide any performance benefit. Apparently the model is not able to learn from these terms without context. The ablation study showed that the model has difficulty to learn correct term boundaries based only on the term lists.

5.3 Limitations and Future Work

A limitation of this study is, that used the used tooling only allowed a flat approach and therefore prevented the detection of nested ICT terms. Furthermore, the annotations of ICT terms were performed by only one person and it was not formally examined whether the annotations remained consistent across the different annotation rounds. The involvement of multiple annotators and an examination of the inter-annotator agreement might help to uncover inconsistencies and further refine the annotation rules.

To further improve the final results, an ensemble approach, combining the strengths of the different computed models could be adopted. Additionally, it could also be investigated to what extent a further enlargement of the training corpus can lead to significant improvements of the model performance.

Moreover, the inclusion of an additional Tok2Vec (token-to-vector) embedding layer to the pipeline might be promising. The *spaCy* Tok2Vec component computes new token based vectors, which can be used by the NER model. For this purpose,a tok2vec model would need to be trained on the job ads.

Further research is also needed on the potential benefit of term lists. This thesis has shown, that an existing list can be expanded very quickly. Within only 2 hours of annotation, the seed term list could be expanded from 2,516 to 6,736 terms. This corresponds to a rate of over 2,000 new terms per hour. Therefore, it would be convenient if such termlists could help to create a NER model for a new entity type. However, it has been shown that the model has difficulty to learn correct term boundaries based on termlists only. Therefore, it could be investigated whether adding some context to the terms help to overcome this problem. Context-based terms could be created relatively easily. For example, when expanding an original term list, an additional window of tokens surrounding an instance of a newly found term, could be stored.

Finally, it remains to be explored to what extent the chosen approach can be transferred to other languages, in particular on the other languages of the Job ad corpus. Outside the context of the NRP project, it would also be interesting to apply the presented approach to other special domain entity types.

6 Conclusion

This thesis has presented an approach to quickly bootstrap a BERT-based NER model for a new, special domain entity type, using the example of ICT term detection in German job ads.

An important first step was the formulation of annotation guidelines for ICT terms. It helped to develop a common understanding of ICT terms together with the project team, but also to ensure a consistent annotation of the training data. Real examples of ICT terms from the advertisements, and in particular unclear cases, helped formalize the concept of "ICT term". However, especially in borderline cases, where the meaning of a term depends on the context, it can be difficult to distinguish ICT terms from non-ICT terms, making it also difficult to formulate general annotation rules. Nested ICT terms present another challenge, which is less conceptual than technical. The used annotation interface of prodigy and the bootstrapped *spaCy* NER model only allow to handle flat terms. Therefore, the annotation of nested terms had to be simplified and only the inner terms were annotated as such. Based on the annotation guidelines, an initial gold standard of 90 ads containing various ICT keywords was created through manual work.

The approach of iteratively increasing training data has proven to be effective. A targeted sampling strategy using a topic model calculated on the entire German job ad corpus, has allowed to train a model that can handle a wide range of ads and ICT terms. With samples of ICT-rich ads, a strongly recall-oriented approach was chosen at the beginning, which made sure that the model was exposed to as many different examples of ICT terms as possible. Later inclusion of samples with less ICT-oriented ads then ensured that the final model could not only handle ads from the ICT domain, but also learned to correctly annotate ads that contain no ICT terms at all, or only a few general or very sector-specific ICT terms. In three rounds of correcting the output of the previous models, within a total of 7 hours, the training material could be increased from 90 to 590 ads. The inclusion of the *prodigy* `ner.correct` interface has contributed to an efficient annotation process. The extension of the training corpus has led to a significant improvement of the model. Evaluated on a development set consisting of 32 ads with different ICT

keywords, the F-score could be increased from 84.36 for the first model, trained only on the 90 manually annotated ads, to 86.79 for the model trained on all 590 ads. At the end, in order to ensure that the final model can handle the whole range of ads as well as possible, a new development set, containing 100 ads from all topics, was created. On this development set, the final model, which was trained on all 590 annotated ads plus the 32 ads from the original development set, finally achieved an F-score of 90.42.

Including term lists with both ICT and non-ICT terms in the training material has not shown a clear benefit to model performance. While these lists are generated very quickly, the model does not seem to be able to learn from individual terms as training data. Further research is needed to investigate whether term lists can provide value under certain circumstances, e.g., by embedding the terms in some context.

The use of domain-specific BERT embeddings, on the other hand, has shown clear benefits. While the very first model, trained on only 90 ads, using domain-specific BERT embeddings already achieved an F-score of 84.36, the same model computed with generic BERT embeddings only reaches an F-Score of 80.73. This is a difference of 3.63 percentage points. By increasing the size of the training material, the difference decreases, but is still there. With an F-score of 89.03, contrary to the final model computed on domain-specific BERT embeddings, the final model computed with generic BERT embeddings could not exceed the 90% limit.

Overall, the chosen approach has proven to be suitable for quickly creating training data and using it to train a NER model for a new special domain entity type. To further improve the results, an ensemble approach combining different models could be adopted. Also the addition of Tok2Vec embeddings to the pipeline might be promising. Furthermore, it remains to be explored to what extent the approach can be transferred to other languages or domains.

References

E. Atalay, P. Phongthiengtham, S. Sotelo, and D. Tannenbaum. The evolution of work in the united states. *American Economic Journal: Applied Economics*, 12 (2):1–34, 2020.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

R. Boselli, M. Cesarini, F. Mercorio, and M. Mezzanzanica. Classifying online job advertisements through machine learning. *Future Generation Computer Systems*, 86:319–328, 2018.

N. Dawson, M.-A. Rizoiu, B. Johnston, and M.-A. Williams. Adaptively selecting occupations to detect skill shortages from online job ads. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1637–1643. IEEE, 2019.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

E. Dierdorff, D. Drewes, and J. Norton. O* net tools and technology: A synopsis of data development procedures. *National Center for O*NET Development, Raleigh, NC*, 2006.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.

S. Fareri, G. Fantoni, F. Chiarello, E. Coli, and A. Binda. Estimating industry 4.0 impact on job profiles and skills using text mining. *Computers in industry*, 118: 103222, 2020.

J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

K. Fort and B. Sagot. Influence of pre-annotation on pos-tagged corpus development. In *The fourth ACL linguistic annotation workshop*, pages 56–63, 2010.

A.-S. Gnehm and S. Clematide. Text zoning and classification for job advertisements in german, french and english. In *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*, pages 83–93, 2020.

Y. Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017.

S. Graham, S. Weingart, and I. Milligan. Getting started with topic modeling and mallet. Technical report, The Programming Historian, 2012.

S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520.

M. Honnibal. Spacy's entity recognition model: incremental parsing with bloom embeddings and residual cnns, Nov 2017. URL `www.youtube.com/watch?v=sqDHBH9IjRU&t=975s`.

D. Jurafsky and J. H. Martin. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3d edition draft edition, 2020.

I. Karakatsanis, W. AlKhader, F. MacCrory, A. Alibasic, M. A. Omar, Z. Aung, and W. L. Woon. Data mining approach to monitoring the requirements of the job market: A case study. *Information Systems*, 65:1–6, 2017.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

T. Lingren, L. Deleger, K. Molnar, H. Zhai, J. Meinzen-Derr, M. Kaiser, L. Stoutenborough, Q. Li, and I. Solti. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial

announcements. *Journal of the American Medical Informatics Association*, 21 (3):406–413, 2014.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

M. Pejic-Bach, T. Bertoncel, M. Meško, and Ž. Krstić. Text mining of industry 4.0 job advertisements. *International journal of information management*, 50: 416–431, 2020.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

J. Pustejovsky and A. Stubbs. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications.* " O'Reilly Media, Inc.", 2012.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

S. Rosset, C. Grouin, T. Lavergne, M. B. Jannet, J. Leixa, O. Galibert, and P. Zweigenbaum. Automatic named entity pre-annotation for out-of-domain human annotation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 168–177, 2013.

X. Schmitt, S. Kubler, J. Robert, M. Papadakis, and Y. LeTraon. A replicable comparison study of ner software: Stanfordnlp, nltk, opennlp, spacy, gate. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 338–343. IEEE, 2019.

SMM. Swiss job market monitor. documentation. scientific use file (release 2020). *Universität Zürich. Distributed by FORS, Lausanne*, 2021. URL `https://forsbase.unil.ch/project/study-public-overview/17480/0/`.

M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

A. Tomar. Topic modeling using latent dirichlet allocation(lda) and gibbs sampling explained!, Nov 2018. URL `https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045`.

Y. Wang, Y. Hou, W. Che, and T. Liu. From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics*, pages 1–20, 2020.

T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer, et al. 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, 2020.

J. Yu, B. Bohnet, and M. Poesio. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*, 2020.

Y. Zhang and J. Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, 2011.

A Tables

Year	SJMM	SJMM-O	OA	Total
2020		5		5
2019	2	5		7
2018	2	5	5	12
2017	2	5	5	12
2016	2	5	5	12
2015	2	5	5	12
2014	1		5	6
2013	1			1
2012	1			1
2011	2			2
2010	2			2
2009	2			2
2008	2			2
2007	2			2
2006	2			2
2005	2			2
2004	2			2
2003	2			2
2002	2			2
2001	2			2
Total	35	30	25	90

Table 14: Distribution of Sample-0 (manually annotated goldstandard) by sources and years. Grey cells: no ads available)

Year	SJMM	SJMM-O	OA	Total
2020		6		6
2019	2	6		8
2018	2	5	6	13
2017	2	5	6	13
2016	2	5	6	13
2015	2	5	6	13
2014	2		6	8
2013	2			2
2012	2			2
2011	2			2
2010	2			2
2009	2			2
2008	2			2
2007	2			2
2006	2			2
2005	2			2
2004	2			2
2003	2			2
2002	2			2
2001	2			2
Total	38	32	30	100

Table 15: Distribution of Sample-1 (ICT ads) and Devset-1 by sources and years. (Grey cells: no ads available)

Year	SJMM	SJMM-O	OA	Total
2020		11		11
2019	4	11		15
2018	4	11	12	27
2017	4	11	12	27
2016	4	10	12	26
2015	4	10	12	26
2014	4		12	16
2013	4			4
2012	4			4
2011	4			4
2010	4			4
2009	4			4
2008	4			4
2007	4			4
2006	4			4
2005	4			4
2004	4			4
2003	4			4
2002	4			4
2001	4			4
Total	76	64	60	200

Table 16: Distribution of Sample-2 (ICT-near ads) and Sample-3 (all topic ads) by sources and years. (Grey cells: no ads available)

Year	SJMM	SJMM-O	OA	Total
2020		6		6
2019	2	6		8
2018	2	5	6	13
2017	2	5	6	13
2016	2	5	6	13
2015	2	5	6	13
2014	2		6	8
2013	2			2
2012	2			2
2011	2			2
2010	2			2
2009	2			2
2008	2			2
2007	2			2
2006	2			2
2005	2			2
2004	2			2
2003	2			2
2002	2			2
2001	2			2
Total	38	32	30	100

Table 17: Distribution of Devset-1 (final development set, based on all topic ads) by sources and years. (Grey cells: no ads available)

Topic	**Excerpt of top 100 words** *Non-ICT words in between are omitted in the enumeration with (...)*
Topic 23	dynamics microsoft kunden ubs bewerben sharepoint business markt chf media bereich arbeiten weltweit gute switzerland crm team mitarbeiter kenntnisse (...) bietet hilti entwickler (...) spezialisiert net marktführer
Topic 30	gute kaufmännische deutsch aufgaben unterstützung administrative vorteil organisation kenntnisse (...) excel ms-office empfang word grundausbildung (...) powerpoint bearbeiten (...) kaufmännischen flexibilität outlook führen assistenz (...) externen anwenderkenntnisse mitarbeit ähnlichen
Topic 31	controlling persönlichkeit human professional personalberatung jahre gute erstellung suchen kundin umfeld (...) sap wünscht vorteil erstellen sitz idealalter gehört accounting führenden arbeitsweise reporting (...) excel ms-office dynamischen
Topic 35	kunden business consultant unternehmen management team data services consulting lösungen (...) weiterbildung finance wirtschaftsinformatik gestalten informationen arbeitsumfeld interesse setzen technologie informatik suchen kenntnisse analyse wissen master
Topic 39	kommunikation medien suchen bereich erfahrung vereinbarung gute aufgaben team kenntnisse schweizer media arbeiten (...) adobe photoshop umsetzung stellen gestaltung verfassen indesign (...) affinität publikationen digitalen externen
Topic 71	marketing bereich umsetzung kommunikation media gute erfahrung digital manager (...) kampagnen erstellung e-commerce englisch jahre weiterentwicklung medien berufserfahrung weiterbildung digitale projekte (...) sales google events seo (...) photoshop
Topic 80	rechnungswesen finanz arbeitgeber buchhaltung kaufmännische gute neuen aufgaben erstellen weiterbildung debitoren bereich vorteil deutsch berufserfahrung (...) sap careerplus ms-office erstellung (...) excel verantwortung abacus grossraum französischkenntnisse umgang
Topic 81	gute entwicklung erfahrung bereich technische kenntnisse erstellen englischkenntnisse technischen kunden maschinenbau (...) automation cad suchen internationalen elektronik (...) software projekte grundausbildung anforderungen (...) mehrjährige programmierung weiterentwicklung umsetzung neuer
Topic 88	gute kunden kaufmännische vorteil deutsch grundausbildung französisch kenntnisse erstellen erfahrung einkauf aufgaben technische bereich verkauf englisch weiterbildung sap lieferanten ausbildung ms-office bearbeitung (...) customer erp überwachen
Topic 92	projektleiter projekte planung suchen erfahrung aufgaben architekten erstellen gute cad weiterbildung ausbildung bau projekten bauleiter kenntnisse architektur bereich kunden vereinbarung (...) autocad führung umsetzung anspruchsvolle ms-office

Table 18: ICT-near topics from Mallet topic model

B Configurations spacy NER model

```
[paths]
train = model-a/train.spacy
dev = model-a/dev.spacy
vectors = null
init_tok2vec = null

[system]
gpu_allocator = "pytorch"
seed = 0

[nlp]
lang = "de"
pipeline = ["transformer","ner"]
batch_size = 128
disabled = []
before_creation = null
after_creation = null
after_pipeline_creation = null
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.ner]
factory = "ner"
moves = null
update_with_oracle_cut_size = 100

[components.ner.model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "ner"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = false
```

```
n0 = null

[components.ner.model.tok2vec]
@architectures = "spacy-transformers.TransformerListener.v1"
grad_factor = 1.0
pooling = {"@layers":"reduce_mean.v1"}
upstream = "*"

[components.transformer]
factory = "transformer"
max_batch_items = 4096
set_extra_annotations = {"@annotation_setters":
    "spacy-transformers.null_annotation_setter.v1"}

[components.transformer.model]
@architectures = "spacy-transformers.TransformerModel.v1"
name = "jobadsBERT/model_sjmm_x28_weighted_addvoc_lbl_ga_20"

[components.transformer.model.get_spans]
@span_getters = "spacy-transformers.strided_spans.v1"
window = 128
stride = 96

[components.transformer.model.tokenizer_config]
use_fast = true

[corpora]

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths.dev}
max_length = 0
gold_preproc = false
limit = 0
augmenter = null

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
max_length = 500
gold_preproc = false
limit = 0
augmenter = null

[training]
```

```
accumulate_gradient = 3
dev_corpus = "corpora.dev"
train_corpus = "corpora.train"
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
dropout = 0.1
patience = 3200
max_epochs = 0
max_steps = 20000
eval_frequency = 200
frozen_components = []
before_to_disk = null

[training.batcher]
@batchers = "spacy.batch_by_padded.v1"
discard_oversize = true
size = 2000
buffer = 256
get_length = null

[training.logger]
@loggers = "spacy.ConsoleLogger.v1"
progress_bar = false

[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
use_averages = false
eps = 0.00000001

[training.optimizer.learn_rate]
@schedules = "warmup_linear.v1"
warmup_steps = 250
total_steps = 20000
initial_rate = 0.00005

[training.score_weights]
ents_per_type = null
ents_f = 1.0
ents_p = 0.0
ents_r = 0.0
```

```
[pretraining]

[initialize]
vectors = null
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
lookups = null
before_init = null
after_init = null

[initialize.components]

[initialize.tokenizer]
```