Universität Zürich[UZH]

Masterarbeit

zur Erlangung des akademischen Grades

Master of Science

der Wirtschaftswissenschaftlichen Fakultät der Universität Zürich

Work Task Classification from Job Ads onto O*NET: Hierarchy-Aware and Cross-lingual Transfer Approach

Verfasserin: Li Jinqiao

Matrikel-Nr: 20-745-063

Referent: Prof. Dr. Martin Volk

Betreuerin: Ann-Sophie Gnehm, Dr. Simon Clematide

Department of Computational Linguistics

Department of Informatics

Submission date: 05.06.2023

Abstract

This project applied a hierarchy-aware and cross-lingual approach to classify job tasks (e.g.: *Verpackungsarbeiten allgemein und in Medizinaltechnik*) from German job advertisements using the ONET English ontology which is a complex ontology with three hierarchical level and fine-grained classes. Two methods, machine translation and multilingual models, are tested to bridge the language gap. The project consisted of two sets of experiments: local classifier experiments using transformer-based models at each hierarchical level, and global hierarchical models on the O*NET data. This work yields several key findings:

Firstly, domain adaptation proved effective, with job domain-specific language models outperforming general domain models. Translation quality also influenced classification performance, with DeepL outperforming the SJMM engine.

Secondly, state-of-the-art models (TextRNN, TextRCNN, HMCN, HiAGM) were used as global hierarchical models for task classification. These models effectively incorporated hierarchical information, addressing inconsistencies and overfitting through recursive regularization.

Furthermore, the best model configurations from both series of experiments are selected to predict job advertisement data, resulting in reliable classification using the O*NET hierarchical ontology. Human post-evaluation, conducted by a German-speaking domain expert, validates the accuracy of the models' predictions. Overall, while this project extensively tested the feasibility of hierarchy-aware classification models, the transformer-based flat model Job-GBERT proves to be a more suitable option for the hierarchical classification of Job Ads data, given its specificity.

Zusammenfassung

In diesem Projekt wurde ein hierarchiebewusster und sprachübergreifender Ansatz zur Klassifizierung von Arbeitsaufgaben (z.B.: *Verpackungsarbeiten allgemein und in der Medizintechnik*) aus deutschen Stellenanzeigen unter Verwendung der englischen ONET Ontologie angewandt, einer komplexen Ontologie mit drei hierarchischen Ebenen und feinkörnigen Klassen. Zwei Methoden, maschinelle Übersetzung und multilinguale Modelle, werden getestet, um die Sprachlücke zu schließen. Das Projekt bestand aus zwei Serien von Experimenten: lokale Klassifikationsexperimente mit transformer-basierten Modellen auf jeder einzelnen Hierarchieebene und globale hierarchische Modelle auf allen O*NET-Daten. Aus dieser Arbeit ergeben sich mehrere wichtige Erkenntnisse:

Erstens erwies sich die Domänenanpassung als wirksam, wobei die jobdomänenspezifischen Sprachmodelle die allgemeinen, domänenunspezifischen Sprachmodelle übertrafen. Auch die Übersetzungsqualität beeinflusste die Klassifizierungsleistung, wobei DeepL die SJMM-Engine übertraf.

Zweitens wurden modernste Modelle (TextRNN, TextRCNN, HMCN, HiAGM) als globale hierarchische Modelle für die Aufgabenklassifizierung verwendet. Diese Modelle berücksichtigten effektiv hierarchische Informationen, indem sie Inkonsistenzen und Überanpassungen durch rekursive Regularisierung beseitigten.

Darüber hinaus werden die besten Modellkonfigurationen aus beiden Versuchsreihen für die Vorhersage von Stellenausschreibungsdaten ausgewählt, was zu einer zuverlässigen Klassifizierung unter Verwendung der hierarchischen Ontologie von O*NET führt. Eine von einem deutschsprachigen Experten durchgeführte Nachevaluierung bestätigt die Genauigkeit der Vorhersagen der Modelle. Insgesamt wurde in diesem Projekt die Machbarkeit hierarchiebezogener Klassifizierungsmodelle umfassend getestet, wobei sich das transformer-basierte flache Klassifikationsmodell Job-GBERT aufgrund seiner Spezifität als geeignetere Option für die hierarchische Klassifizierung von Stellenanzeigendaten erweist.

Acknowledgement

I would like to express my gratitude to the following individuals who provided invaluable assistance and guidance throughout the course of this project:

First and foremost, I am sincerely thankful to Prof. Martin Volk for granting me the opportunity to undertake my thesis in the Computational Linguistics Department. His support and arrangement are instrumental in shaping the direction of this research. I would also like to extend my appreciation to Dr. Simon Clematide for his assistance in resolving the GPU-related challenges encountered during the project.

A special mention goes to Ann-Sophie Gnehm, whose unwavering support and practical guidance are invaluable. Her professionalism, patience, and dedication are evident in our regular meetings, and her meticulous suggestions greatly contributed to various aspects of the project, including experiment setup, coding, and report writing. Her valuable insights and discussions ensure the timely completion of this work.

Lastly, I am deeply grateful to my parents for their unwavering support and encouragement throughout my academic journey. Their financial and moral backing provided me with the peace of mind necessary to dedicate three years to my postgraduate studies in Zurich.

Thank you all for your invaluable contributions and assistance, without which this project would not have been possible.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the labour market, job advertisement is the first connection between candidates and human resources professionals. Human resources professionals use job advertisements to find new hires, and candidates apply for a vacant position through a job advertisement. Usually, job advertisement text is written in a highly standardized way, it contains sufficient and certain information about the background and value of the company; requirements, tasks and benefit sharing for candidates; process of application and contact information of the talent acquisition. Thus, from an IT perspective, this job advertisement information can be retrieved and processed within an automatic system, in order to serve the labour market. For example, by analyzing the data, the authority can know which field is the most trending one; job seekers can find which skills are the most desired in the market, and human resources professionals can design job listings as they need, etc.

The Swiss Job Market Monitor (SJMM)[1] is an extensive database of job advertisements with data dating from 1950 to the present. This system is dedicated to the systematic monitoring and forward-looking analysis of the Swiss labour market. Currently, the data collection of SJMM comes mainly from panel of job portals and the corporate panel. Since 2006, the automated software has been collecting data from job portals continuously. Now it stores more than 1.3 million new ads per year, which ensures that 95% of the ads posted on Swiss websites are covered. At the same time, the database collects around 4,700 ads per quarter from over 1,300 company websites. Since 2001, the database collection of open jobs has been extended from the German-speaking region to the whole of Switzerland. (Therefore, the corpus languages, include several languages that are widely spoken in Switzerland: English, German, French, Italian, and Romance languages). The database provides the Scientific Use File (SUF) as a database for various academic studies on long-term trends in the Swiss labour market.

[1] `https://www.stellenmarktmonitor.uzh.ch`

Some research work for automated structuring and processing based on SJMM has already been done. **Text zoning** solution can be the starting point to achieve this automation process. Based on Gnehm's previous work on extraction, text zoning has been implemented using neural networks to identify different paragraphs in the advertisements, such as object, task, activity, etc. Furthermore, **Named Entity Recognition (NER)** style extraction is applied for information and communications technology (ICT) terms to explore required skills (Gnehm et al. [2022b]). Since tasks are the essential and practical section in a job ad to explain the position content, classifying tasks into some standardized categories is also an important step. Gnehm et al. extracted the skills in job advertisements and classified into the *European Skills, Competences, Qualifications and Occupations*(ESCO) ontology.

Kind of similar to Gnehm et al.'s previous classification, in this project, I plan to classify the *job task* into O*NET work activities ontology. The *task* is one of the most important parts of a job ad, and is where the company describes what the job actually is, lists responsibilities and gives insight into what the candidate is expected to do in the job. The task description usually makes up a large part of the job ad text and can be formulated as running text, but is often also in a list format.

The Occupational Information Network (O*NET) is designed as a tool for career exploration and job analysis that facing to the full market. It offers three O*NET work activities and task statements, which can be used in this study as the standardized categories or hierarchical ontology data for classifying job ad tasks. The O*NET ontology data contains task statements and a three-level hierarchy for work activity categories. Around 18,000 task statements for professions are assigned to one or more Detailed Work Activities (DWA) which is the lowest level in the hierarchy. Besides the DWA, there are around 300 Intermediate Work Activities (IWA) and around 40 Generalized Work Activities (GWA).

Compared to the conventional flat classification approach, which flattens hierarchical data and feeds it into machine translation models, employing hierarchical classification for job advertisements offers several potential advantages:

- **Data fitting**: Job advertisements exhibit significant variations in the level of detail within task descriptions due to the large number of ads and diverse sources. Hierarchical classification enables the assignment of both detailed and general task descriptions to appropriate categories, accommodating the non-uniform nature of the data.
- **Information gain**: Previous research by Gnehm (2022) in skill classification demonstrated improved results when considering parent-class labels. Hence,

leveraging the hierarchical structure of the ontology is expected to yield performance benefits by capturing valuable information and relationships between categories.

- **Consistency in labels**: Utilizing a single, relatively complex model that considers the entire class hierarchy as a whole offers an appealing alternative to employing multiple local classifiers (e.g., per node or per level). Implementing several classifiers introduces challenges during implementation, such as addressing the disregard of parent-child relationships in methods like Local Classifier per Level (LCL). This omission potentially leads to a significant loss of knowledge that the classification model could otherwise learn. Moreover, the use of multiple classifiers may result in inconsistent categorizations for the same task.

By employing hierarchical classification for job advertisements, these advantages can be harnessed to enhance the accuracy, flexibility, and overall performance of the classification process.

To explore the performance of the global model, we consider testing several newly proposed methods in this study. Neural networks are widely used to solve hierarchical classification problems. In this project, I am going to test TextRNN, TextRCNN and Hierarchical Multi-label Classification Network (HMCN). Another approach was proposed in 2020, and the **HiAGM** (Zhou et al. [2020]) is a robust global hierarchical perception model that outperforms other models in many public datasets.

As the ontology data in O*NET is in English and the majority of the job advertisement data in our corpus is in German, the language gap is another problem that cannot be neglected. Therefore, the second aim of this study will address the language gap between the Job Ad data and O*NET ontology data. In this study, I am going to compare the performance of traditional machine learning methods with transformer-based multilingual models.

1.2 Research Questions

The research questions for this study are as follows:

1. **Assessing the suitability of hierarchical text classification approaches**: What is the effectiveness of hierarchical text classification approaches for the task of classifying tasks from job advertisements into the O*NET hierarchical ontology data? This research question examines the efficacy of hierarchical

text classification methods in categorizing job advertisement tasks into the O*NET hierarchical ontology. Baseline models with flatten classifiers will be established as a starting point, and their performance will be compared against models that incorporate semantic similarity measures between classes and textual data. Specifically, the approaches proposed by Zhou et al. [2020] and Wehrmann et al. [2018] will be implemented and evaluated.

2. **Addressing the language gap between job advertisements and ontology data**: How can the language gap between job advertisements and ontology data be effectively addressed? This research question focuses on exploring strategies to overcome the language disparity between job advertisements and ontology data. Two approaches will be tested and compared: machine translation and multilingual models. For the machine translation approach, various techniques, such as the SJMM domain-adapted machine translation system and the DeepL API, will be employed to translate the ontology data into German. The translated ontology data will then be utilized to train a model for classifying German job advertisement task data. Alternatively, multilingual modeling techniques, utilizing models like XML-RoBERTa-base, will be trained directly on English ontology data to classify German job advertisement tasks.

Through the investigation of these research questions, this study aims to determine the effectiveness of hierarchical text classification approaches and identify the most suitable approach for bridging the language gap between Job Ads data and O*NET ontology data.

1.3 Thesis Structure

The remaining content of this report consists of five chapters:

Chapter 2 presents an overview of relevant previous studies related to the project. This includes discussions on hierarchical text classification, recursive regularization, and machine translation. It also explores various global hierarchical models such as HMCN, HiAGM, and their variants.

Chapter 3 provides a comprehensive description of the SJMM and O*NET datasets. Basic dataset analysis is presented, along with an explanation of the methods employed in the project.

Chapter 4 outlines the experiment design in detail. This chapter covers the ex-

perimental setup, data processing techniques, experiment flow, and the evaluation metrics utilized in the project.

Chapter 5 presents a thorough analysis of the experimental results. It includes visual analysis, explanations of the results, and detailed examples for better comprehension.

Chapter 6 concludes the report by summarizing the key findings of this work. Additionally, it highlights areas for improvement and suggests future directions for further research.

By organizing the report into these chapters, a comprehensive understanding of the project, related studies, methodology, experimental results, and conclusions can be achieved.

2 Related work

2.1 Hierarchical Text Classification (HTC)

Hierarchical Text Classification (HTC) is an important but challenging subtask of Text Classification (TC), which is a common real-world problem that happens when the targets of the classification task are organized in a hierarchical order instead of flat. The taxonomic hierarchy is commonly modeled as a tree or a directed acyclic graph, where each label is regarded as a node that may have children and can be classified. Figure 2.1 is a visualized example from the O*NET dataset. The task *'Review and analyze legislation, laws, or public policy and recommend changes to promote or support interests of the general population or special group.'* is the input text of the model and then it will be assigned to labels at three different hierarchical levels. At the highest GWA level, this task is assigned to two classes *'Analyzing Data or Information'* and *'Providing Consultation and Advice to Others'*, similarly in the fine-tuned IWA and GWA levels.

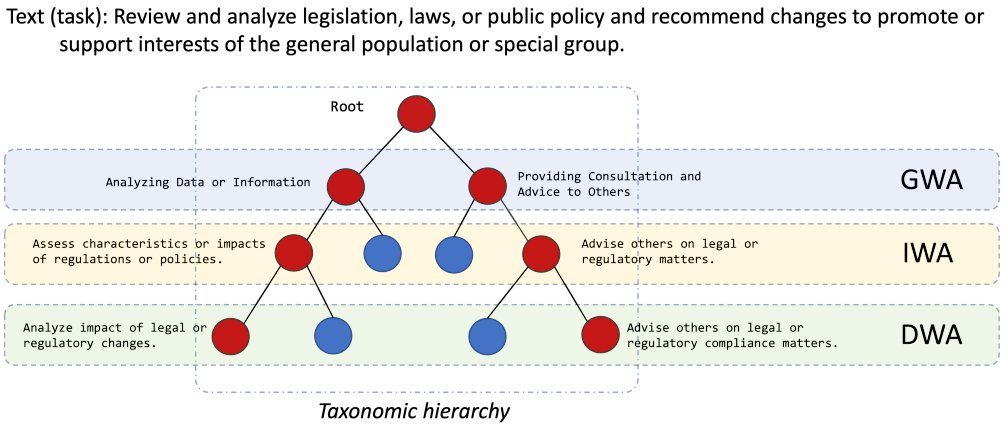


Figure 2.1: An example of hierarchical text classification task.

More particularly, HTC is a specific type of the Hierarchical Multilabel Classification (HMC). Which means that if a label is classified as a child node, it's also belonging to the respective parent nodes. In figure 2.1, the input text is assigned to the label *'Analyze impact of legal or regulatory changes'* at the lowest DWA level. So, it also

classified to the corresponding parent labels at IWA and GWA levels respectively.

HMC problem is defined by Vens et al. [2008] as whose samples satisfy i) may belong to multiple classes simultaneously and ii) organized within a hierarchy. The applications can be in many domains, like text classification (Rousu et al. [2006]), image annotation (Dimitrovski et al. [2011]), and protein function prediction (Otero et al. [2010]).

The existing approaches for HTC problems are commonly categorized as local or global approaches. The local approaches usually construct multiple classifiers for particular nodes or particular hierarchical levels. The local studies follow the top-down strategy to generate hierarchy and overcome the data imbalance on child nodes by learning from parent nodes (Wehrmann et al. [2018]). On the other hand, the global approach uses single classifier for the whole hierarchy structure. Unlikely to the early global approaches which treat the HTC as a flat multi-label classification problem (Johnson and Zhang [2015]), some global studies take advantage of hierarchical structure with specific strategies:

- **Recursive regularization:** Gopal and Yang proposed to combine the labels' hierarchical dependencies with the regularization structure of the parameters. In this way, they encourage classes nearby in the hierarchy to share similar model parameters.
- **Meta learning:** Wu et al. applies a meta-learner to jointly learn the training and prediction policies for different labels. Then, the training policies are used to train the classifier with the cross-entropy loss function, and the prediction policies are used for prediction process.
- **Reinforcement learning:** Mao et al. consider HTC as a Markov decision process. They proposed the Hierarchical Label Assignment Policy (HiLAP) model via deep reinforcement learning to determine where to place an object and when to stop the assignment process.

Both local and global approaches have their own advantages and disadvantages. Typically, local models are more prone to exposure bias as they lack holistic structural information (Zhou et al. [2020]). However, they are adept at extracting information from different regions of the class hierarchy, until overfitting occurs. On the other hand, global approaches are less likely to suffer from the error-propagation problem, but are less capable of capturing local information, leading to underfitting (Wehrmann et al. [2018]).

There are some popular data sets that are often used to test models' performance

on HMC tasks.

1. **Reuters Corpus Volume I (RCV1)** [1] is a benchmark dataset on text categorization. It is a collection of newswire articles produced by Reuters in 1996-1997. It contains 804,414 manually labeled newswire documents and is categorized with respect to three controlled vocabularies: industries, topics and regions (Lewis et al. [2004]).

2. **The Web of Science (WOS) Dataset** [2] is a collection of data of published papers available from the Web of Science. WOS has been released in three versions: WOS-46985, WOS-11967 and WOS-5736. WOS-46985 is the full dataset. WOS-11967 and WOS-5736 are two subsets of WOS-46985.
   For example, WOS-11967 contains 11,967 documents with 35 categories which include 7 parent categories. *Domain* is the root hierarchical level which includes 7 labels: Computer Science, Electrical Engineering, Psychology, Mechanical Engineering, Civil Engineering, Medical Science, Biochemistry. *Area* is subdomain or area of the paper such as computer graphics (Kowsari et al. [2017])

3. **The New York Times (NYT) Annotated Corpus** [3] contains over 1.8 million articles written and published by the New York Times between January 1, 1987 and June 19, 2007 with article metadata provided by the New York Times Newsroom, the New York Times Indexing Service and the online production staff at nytimes.com.

Data Set	$\|L\|$	Depth	Avg($\|L_i\|$)	Train	Val	Test
RCV1-V2	103	4	3.24	20833	2316	781265
WOS	141	2	2	30070	7518	9397
NYT	166	8	7.6	23,345	5,834	7,292

Table 2.1: Datasets Statistics. $|L|$ is the number of classes. Depth is the maximum level of hierarchy. Avg($|L_i|$) is the average number of classes per sample. Train/Val/Test are the size of train/validation/test set respectively.

In 2018, Wehrmann et al. combined local and global optimizations in a hybrid model called HMCN. This neural network architecture is specifically designed for HMC problem. It can discover local hierarchical class relationships and capture the entire class hierarchy while penalizing hierarchical violations.

[1] `http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm`

[2] `https://data.mendeley.com/datasets/9rw3vkcfy4/6`

[3] `https://catalog.ldc.upenn.edu/LDC2008T19`

Some recent studies state that a structure encoder demonstrating the taxonomy hierarchy structure directly can further improve the global models' performance. In 2020, Zhou et al. design an encoder that integrates the label prior probability to learn label hierarchy representations. Based on the hierarchy encoder, they propose a hierarchy-aware global model (HiAGM) with two variants: a multi-label attention variant (HiAGM-LA) and a text feature propagation model (HiAGM-TP).

Some further study conducted based on the HiAGM. Chen et al. embeds word and label jointly in the hyperbolic space by considering this task as semantic matching and applying BERT as encoder. Wang et al. propose Hierarchy-guided Contrastive Learning (HGCLR) to embed the hierarchy into a text encoder directly. They train the model constructively by constructing positive samples, so that the text encoder can dispense with the redundant hierarchy.

Model	RCV1-V2		WOS		NYT	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
HMCN (Mao et al. [2019])	80.80	54.60	-	-	-	-
TextRCNN (Zhou et al. [2020])	81.57	59.25	83.55	76.99	70.83	56.18
HiAGM (Zhou et al. [2020])	83.96	63.35	85.82	80.28	74.97	60.83
HiMatch Chen et al. [2021]	84.73	64.11	86.20	80.53	-	-

Table 2.2: Some experiment results of recent models on three public datasets.

In this project, we are going to apply some of these latest effective models to O*NET ontology data and compare their performance on the job ads classification task.

2.2 Cross-lingual Transfer Approaches

Cross-lingual transfer learning aims to use models and resources in one language and transfer them to a different language. In practice, collecting annotated data for all targeted languages is too expensive and time-consuming. With the cross-lingual transfer learning, it is possible for models to learn with the high-resource language and then transferring it to other low-resource languages.

2.2.1 Multilingual Models

Some large pre-trained multilingual neural language models, like **Multilingual BERT** (m-BERT) and **XLM-RoBERTa** (Conneau et al. [2020]) have been shown to generalize in a zero-shot cross-lingual setting. This generalization ability has been

attributed to the use of a shared subword vocabulary and joint training across multiple languages giving rise to deep multilingual abstractions (Artetxe et al. [2020]).

Cross-lingual transfer utilizes large pre-trained multilingual transformer models which are fine-tuned with training data in a high-resource language and then used to predict entries from other languages than that used in training, often with satisfying results (Pires et al. [2019], Wu and Dredze [2019]). As such, the current consensus of the cross-lingual generalization ability of mBERT is based on a combination of three factors: i) shared vocabulary items that act as anchor points; ii) joint training across multiple languages that spreads this effect; which ultimately yields iii) deep cross-lingual representations that generalize across languages and tasks (Artetxe et al. [2020]).

2.2.2 Machine Translation

Recent studies in Machine Translation (MT) have brought forth a new paradigm for building NLP applications in low-resource scenarios. To build a classifier for a language with no labeled resources, one can translate labeled data from another language, then train a classifier on the translated text. This can be viewed as a domain adaptation problem, where labeled translations and test data have some mismatch (Duh et al. [2011]).

Early studies from Wan, Mihalcea et al. address the Multilingual classification problem using cross-lingual training with machine translation strategy. Instead of using machine translation engines to translate labeled text, Blitzer et al. use it to construct the word translation oracle for pivot words translation.

However, Artetxe et al. also claimed that domain mismatch was not caused by machine translation (MT) errors, and accuracy degradation would occur even with perfect MT.

3 Data & Methods

3.1 Job Ad Data

The job advertisement data is originally from the SJMM database. The data I used for this project is based on Gnehm and Clematide [2020]'s previous work.

sample	id	text	label
1	sjmm-22011109412481	Verpackungsarbeiten allgemein und in Medizinaltechnik	ActObjCont
2	sjmm-22011109412481	Einrichten des eigenen Arbeit-splatzes	
3	sjmm-22011108302017	geophysikalischen Messungen (Gravimetrie, Bohrlochgeophysik	
4	sjmm-22011108302017	Erarbeitung und Umsetzung von Explorations- / Untersuchungs konzepten und Arbeitsprogrammen	
5	sjmm-22011108302017	Planung und Durchführung von Forschungsprojekten in Felslabors	

Table 3.1: Examples of job ad data (with context).

Table 3.1 gives an overview of what the *task* in the Job Ads dataset looks like. In this table, we can see that one job advertisement may contain several job tasks. In this table, sample 1 and sample 2 are from the same advertisement because they have identical *id* and samples 3,4,5 are from another one.

In Table 3.1, *'id'* is the unique number of the job ad in the SJMM database. *'label'* is used to indicate whether the text is with or without context. *'text'* refers to the 'task' which is the most important and only used part in this work.

As Gnehm and Clematide's description, the *task* is the extracted combination of Object and Activity of one advertisement. In order to get the text spans (like *task*) from the job ads, the authors first process the whole job ad texts with text zoning into 8 different zones, e.g., the company description, skill requirements, job task description, administrative parts, etc (Gnehm et al. [2022b]). Then, the next step is similar to how the skills are extracted (Gnehm et al. [2022a]), where the authors have a NER-style extraction of single job tasks. However, it is even more fine-grained,

because it finally gets into single aspects or subspans of this job tasks, like Activity, Objects, Responsibility Role and so on.

In this project, I work with a combination of the subspans, that are considered the most relevant for the classification onto O*NET, namely the Activities, Objects, ObjectActivities and Context.

Then, for all the job ads *tasks* with context (size: 4586), I plot the text length distribution (see Figure 3.2). We can see that most data lengths are concentrated in the range of 1 to 15 words. Table 3.2 gives more statistics details about the job ads text length distribution. The average length in this data set is 5.6339 words, which is a very short number compared to the text in the training O*NET dataset. In the O*NET dataset, the average length for original English text is 13.82. For SJMM and DeepL translated German data, these numbers are 14.2 and 14.9, respectively.

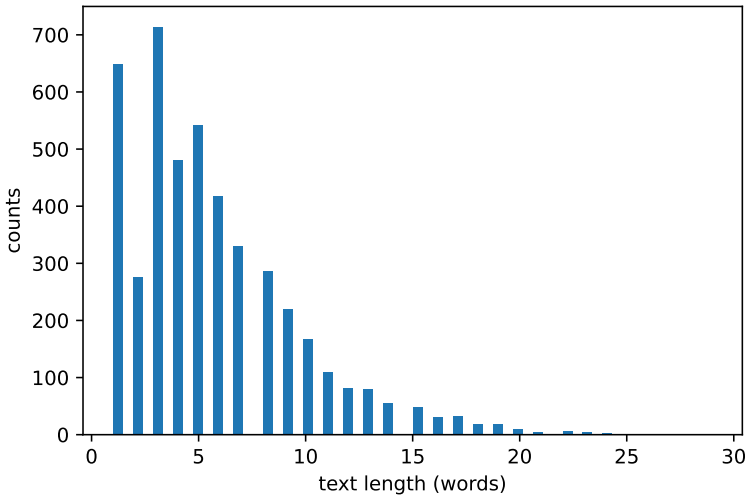


Figure 3.1: Job ads tasks length distribution

	mean	std	min	25%	50%	75%	max
length	5.6339	3.9836	1	3	5	8	29

Table 3.2: Statistics description of job ads text length distribution.

3.2 O*NET Data

Data set is available on Occupational Information Network (O*NET) [1] which is a digital database containing occupational characteristics and worker requirements

[1] `https://www.onetcenter.org/database.html`

information about American labor market. In this work, we use the collected *TASK* and *Work Activities* data from The O*NET 27.2 Database.

In the Task Statement data file, there are eight columns for each row named O*NET-SOC Code, Title, Task ID, Task, Task Type, Incumbents Responding, Date, and Domain Source (see Fig 3.2).

O*NET-SOC Code	Title	Task ID	Task	Task Type	Incumbents Responding	Date	Domain Source
41-9031.00	Sales Engineers	9682	Develop, present, or respond to proposals for specific customer requirements, including request for proposal responses and industry-specific solutions.	Core	47	08/2022	Incumbent
41-9031.00	Sales Engineers	9680	Collaborate with sales teams to understand customer requirements, to promote the sale of company products, and to provide sales support.	Core	49	08/2022	Incumbent
41-9031.00	Sales Engineers	9690	Create sales or service contracts for products or services.	Core	49	08/2022	Incumbent
41-9031.00	Sales Engineers	9689	Visit prospective buyers at commercial, industrial, or other establishments to show samples or catalogs, and to inform them about product pricing, availability, and advantages.	Core	47	08/2022	Incumbent
41-9031.00	Sales Engineers	9692	Keep informed on industry news and trends, products, services, competitors, relevant information about legacy, existing, and emerging technologies, and the latest product-line developments.	Core	47	08/2022	Incumbent

Figure 3.2: Data Example - Task Statements

This database contains three different levels of job activities:

The Work Activities data is the hierarchical ontology that is considered as labels for each task. There are three different levels of job activities from coarse-grained to fine-grained: Generalized Work Activities (GWA), Intermediate Work Activities (IWA), and Detailed Work Activities (DWA).

- 41 Generalized work activities (GWA):
  Analyzing Data or Information / Working with Computers / Selling or Influencing Others / ...
- 332 Intermediate work activities (IWA):
  Analyzing Data or Information: Analyze environmental or geospatial data. / Analyze business or financial risks. / ...
- 2,069 Detailed work activities (DWA):
  Analyze business or financial risks: Assess risks to business operations. / Analyze risks related to investments in green technology. / Analyze risks to minimize losses or damages. / ...

18,000 task statements that are assigned to one or more DWAs. For example, task *Review prescriptions to assure accuracy, to ascertain the needed ingredients, and to evaluate their suitability (ID: 1808)* is only assigned to *Verify accuracy of patient information.* However the task *Provide information and advice regarding drug interactions, side effects, dosage, and proper medication storage.(ID: 1809)* is

assigned to both *Communicate detailed medical information to patients or family members* and *Advise patients on effects of health conditions or treatments.* Same as task 1809.

Data Example - Tasks to DWAs:

O*NET-SOC Code	Title	Task ID	Task	DWA ID	DWA Title	Date	Domain Source
29-1051.00	Pharmacists	1808	Review prescriptions to assure accuracy, to ascertain the needed ingredients, and to evaluate their suitability.	4.A.2.a.2.I05.D04	Verify accuracy of patient information.	03/2014	Analyst
29-1051.00	Pharmacists	1809	Provide information and advice regarding drug interactions, side effects, dosage, and proper medication storage.	4.A.4.a.1.I05.D01	Communicate detailed medical information to patients or family members.	03/2014	Analyst
29-1051.00	Pharmacists	1809	Provide information and advice regarding drug interactions, side effects, dosage, and proper medication storage.	4.A.4.b.6.I01.D04	Advise patients on effects of health conditions or treatments.	03/2014	Analyst
29-1051.00	Pharmacists	1811	Order and purchase pharmaceutical supplies, medical supplies, or drugs, maintaining stock and storing and handling it properly.	4.A.4.c.3.I01.D02	Maintain inventory of medical supplies or equipment.	03/2014	Analyst
29-1051.00	Pharmacists	1811	Order and purchase pharmaceutical supplies, medical supplies, or drugs, maintaining stock and storing and handling it properly.	4.A.4.c.3.I05.D07	Order medical supplies or equipment.	03/2014	Analyst

Figure 3.3: Data Example - Task to DWAs

Task	Review and analyze legislation, laws, or public policy and recommend changes to promote or support interests of the general population or special group
DWA	Direct financial operations
IWA	Manage budgets or finances
GWA	Guiding, Directing, and Motivating Subordinates
Occupation	Chief Executives

Table 3.3: An example in Task Dictionary

Figure 3.4 shows the original English text length distribution where most data lengths are concentrated in the range of 2 to 30 words. Table 3.7 gives more statistics details of tasks about the task length distribution for three different versions.

3.2.1 Tasks Distribution

There are a number of different categories at each of the three levels of the classification hierarchy, and each category contains a different number of tasks. The imbalance of task distribution among different classes and the sparse data would make the work harder. Therefore, I visualized the distribution of tasks across the different levels to get a whole picture of the data and also to understand the difficulty of the task better.

Figure 3.5 is the distribution of tasks on the level of DWA, the lowest level in the

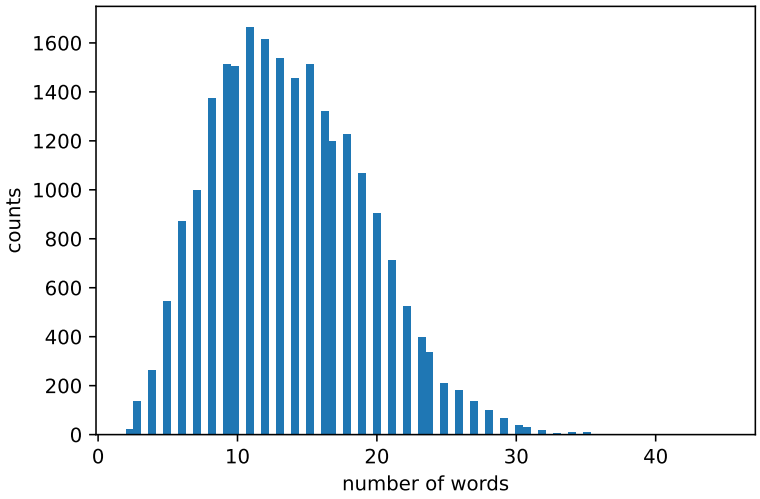


Figure 3.4: Text length distribution of tasks (English).

classification hierarchy. This plot shows that on the DWA level, most categories contain 1-20 tasks, but a few categories contain up to 122 tasks.

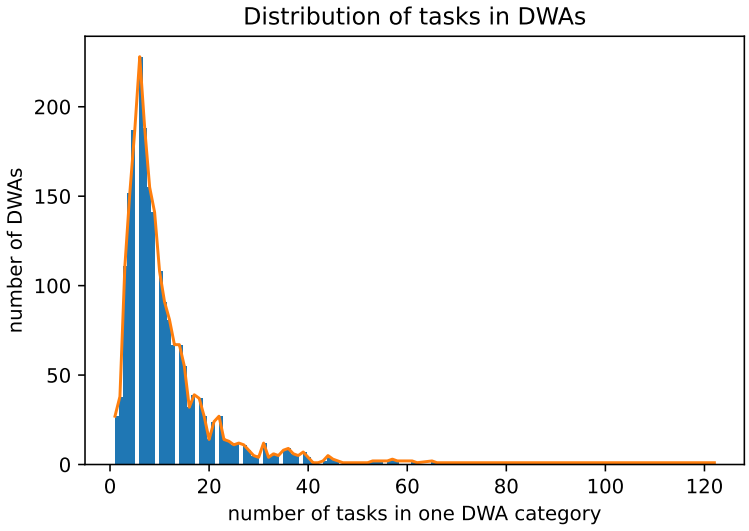


Figure 3.5: Distribution of tasks in the level of DWA.

Following Table 3.4 provides the statistics details of tasks distribution on the three different hierarchy levels.

Level	mean	std	min	25%	50%	75%	max
GWA	636.30	604.69	44	168	395	800	3055
IWA	70.91	68.66	3	28	52	87	545
DWA	11.29	10.37	1	6	8	13	122

Table 3.4: Statistics description of tasks distribution at different classification levels.

3.2.2 Multilabel Situation

One of the biggest features of this project is that this is a multilabel classification problem which means in the O*NET data set, one task may have more than one label.

I counted the task multilabel distribution situation on each hierarchical level, as in Table 3.5. In the GWA level, 2723 tasks have two activity labels and 211 tasks have three labels. While in the IWA level, other than 3150 and 374 tasks are assigned to two and three labels respectively, 5 tasks are assigned to four IWA classes. The situation is similar in the fine-grained DWA level, only the number of max activities assigned to one task peaks at 5. From the highest GWA level to the fine-grained DWA level, the percentage of multilabel increases from 13.35% to 17.33%. This is understandable, in Table 3.6, five different DWAs are assigned to task 22935, while only 4 and 3 activities are assigned to it on the IWA and GWA levels respectively.

Level	Categories	Percent_multi (%)	1	2	3	4	5
GWA	37	13.35%	15897	2723	211	-	-
IWA	332	15.52%	15302	3150	374	5	-
DWA	2085	17.33%	14752	3453	620	5	1

Table 3.5: Details at different classification levels. 1-5 is the number of class one task belongs to on a certain level. 'Percent_multi' is the percent of tasks have more than one label.

Task 22935 *'Analyze burn conditions and results, and prepare postburn reports'* has the most labels in the O*NET data set, the only task which has 5 different DWA labels. Table 3.6 gives the details of it. The task 22935 is assigned to five different DWA labels and two of the DWA labels belong to the same IWA class while others do not. Similarly, these four IWA labels come from three different GWA classes.

Task	DWA	IWA	GWA
(22935) Analyze burn conditions and results, and prepare postburn reports.	(4.A.1.a.2.I07.D10) Locate fires or fire danger areas.	(4.A.1.a.2.I07) Monitor safety or security of work areas, facilities, or properties.	(4.A.1.a.2) Monitoring Processes, Materials, or Surroundings
	(4.A.1.a.2.I09.D02) Monitor environmental conditions to detect hazards.	(4.A.1.a.2.I09) Monitor environmental conditions	
	(4.A.1.a.2.I09.D05) Assess characteristics of fires.		
	(4.A.3.a.1.I02.D03) Perform forest fire-fighting activities.	(4.A.1.a.1.I02) Protect people or property from threats such as fires or flooding.	(4.A.3.a.1) Performing General Physical Activities
	(4.A.3.b.6.I15.D04) Prepare operational reports.	(4.A.3.b.6.I15) Prepare reports of operational or procedural activities.	(4.A.3.b.6) Documenting/Recording Information

Table 3.6: An example for multi-labels: labels of task (ID: 22935) at different hierarchy levels.

3.2.3 Machine Translation

To address the language gap between the job ads and O*NET ontology data, the first solution we tried is to translate the ontology data from English to German. At this step, we test two different machine translation methods.

- **SJMM translation** This is a task-specification translation model proposed by Magaldi for job advertisements of the Swiss Job Market Monitor. This model is trained with very low amount of in-domain parallel data and supported by fine-tuning with out-of-domain data.
- **DeepL translation:** Using the DeepL API [2] for python, the original ontology data is translated to German version.

I compared the text length of original English tasks and two translated German version.

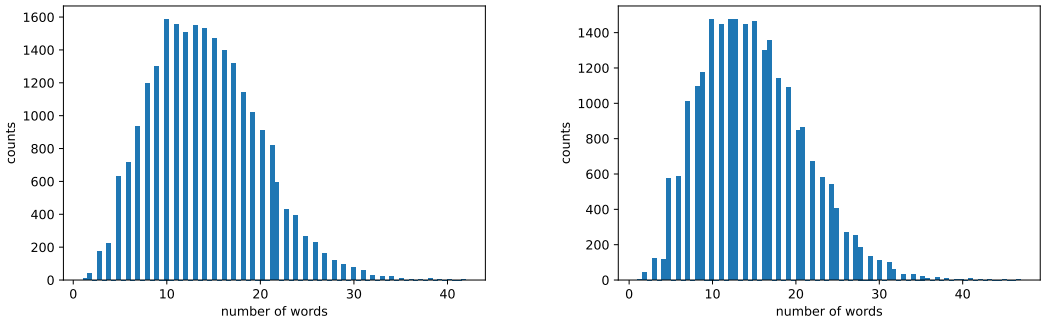
Following is an example of the translation in two ways, including the the text of task and its corresponding labels in three different levels.

In this example, we can find out that the translated German version for ontology

[2] `https://github.com/DeepLcom/deepl-python`

Level	mean	std	min	25%	50%	75%	max
English	13.82	5.50	2	10	13	18	45
SJMM	14.20	5.75	1	10	14	18	42
DeepL	14.90	6.12	1	10	14	19	47

Table 3.7: Statistics description of tasks length (word) with different translation system



(a) SJMM (b) DeepL

Figure 3.6: German tasks length distribution

	English	German_SJMM	German_DeepL
Task	Review and analyze legislation, laws, or public policy and recommend changes to promote or support interests of the general population or special groups	Überprüfung und Analyse von Rechtsvorschriften, Gesetzen oder der öffentlichen Politik und Empfehlung von Änderungen zur Förderung oder Unterstützung der Interessen der Allgemeinheit oder bestimmter Gruppen.	Überprüfung und Analyse von Rechtsvorschriften, Gesetzen oder der öffentlichen Ordnung und Empfehlung von Änderungen zur Förderung oder Unterstützung der Interessen der allgemeinen Bevölkerung oder spezieller Gruppen.
GWA	Analyzing Data or Information	Analysieren von Daten oder Informationen	Analysieren von Daten oder Informationen
IWA	Assess characteristics or impacts of regulations or policies.	Beurteilung von Merkmalen oder Auswirkungen von Verordnungen oder Polizisten.	Bewertung von Merkmalen oder Auswirkungen von Vorschriften oder Politiken.
DWA	Analyze impact of legal or regulatory changes.	Analyse der Auswirkungen von rechtlichen oder regulatorischen Veränderungen.	Analyse der Auswirkungen von rechtlichen oder regulatorischen Veränderungen.

Table 3.8: An examples of machine translation on ontology data

labels are exactly same with two methods. There is only some difference in the text of task. After examing other tasks in data set, I found this situation is very common. This is reasonable, because the length of ontology text is much shorter compared with task text.

3.3 Methods

3.3.1 Neural Networks

It is a common way to solve NLP problems with neural models. In this part, I mainly explain the **R**ecurrent **N**eural **N**etwork (RNN) and **R**ecurrent **C**onvolutional **N**eural **N**etworks (RCNN) models, because these two models play important roles in the following experiments. By combining with recursive regularization, these models can also be applied to the hierarchical text classification problem.

1. **RNN**
   RNN is one of the most popular architecture used in NLP problems because their recurrent structure is very suitable to process the variable-length text.

   As mentioned in section 2.1, hierarchical text classification is a special case of the multi-label classification problem. In 2016, Liu et al. proposed to solve text classification with RNNs using Multi-Task learning. In their work, based on recurrent neural networks, they created three different multi-task learning mechanisms that learn jointly across multiple related tasks. Their work first integrates RNNs into a multi-task learning framework to learn to map arbitrary text to semantic vector representations with specific tasks and shared layers.

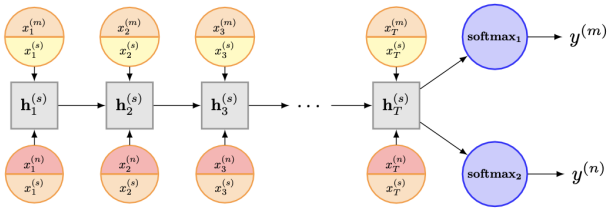
   Model-I is a single LSTM model and all tasks share the same LSTM layer. For each input character, a trainable vector is concatenated after the embedding vector to represent the particular task. And the last moment of the hidden state is passed as input to the softmax and results for the different tasks.

   $$\hat{x_t} = x_t^{(m)} \oplus x_t^{(s)}$$

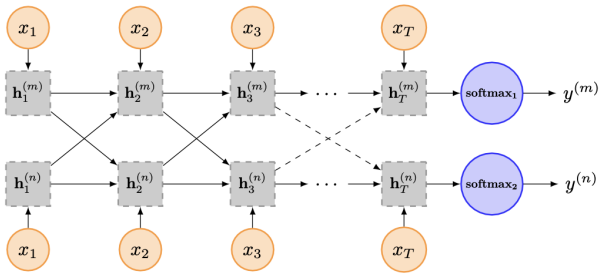
   where $x_t^{(m)}$ denotes the specific task embedding, and $x_t^{(s)}$ denotes the shared word embedding, $\oplus$ denotes the concatenation operation.

   Model-II contains a two-layer LSTM and assigns one LSTM for each task. To connect the two LSTM models, there is a gate between each cell in the LSTM models to decide how much information it can accept from another task model. The new memory content in the LSTM layer at m-th task is re-defined as follow:

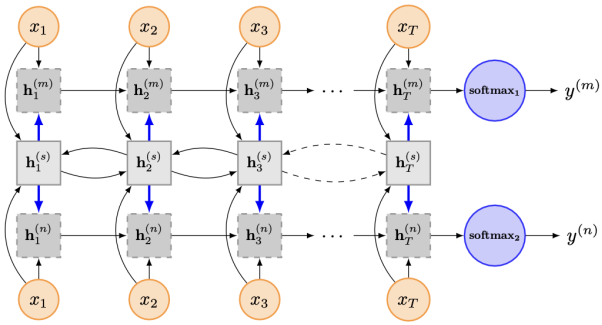
   $$\tilde{c}_t^{(m)} = \tanh(W_c^{(m)} x_t + \sum_{i \in \{m,n\}} g^{(i \to m)} U_c^{(i \to m)} h_{t-1}^{(i)})$$



(a) Model-I: Uniform-Layer Architecture



(b) Model-II: Coupled-Layer Architecture



(c) Model-III: Shared-Layer Architecture

Figure 3.7: Three architectures for modelling text with multi- task learning (Liu et al. [2016]).

Model-III Besides a separate LSTM layer for each task, there is also a shared bidirectional LSTM layer to capture the global information of all the tasks. The output of step t is denoted as $h_t^{(s)} = \overrightarrow{h}_t^{(s)} \oplus \overleftarrow{h}_t^{(s)}$, where $\overrightarrow{h}_t^{(s)}$ and $\overleftarrow{h}_t^{(s)}$ denote the outputs of forward and backword respectively.

2. **RCNN**

   Lai et al. [2015] firstly proposed the RCNN for text classification. A CNN model is usually composed of a convolutional layer and a pooling layer. However, in their work, the convolutional layer is replaced by a bidirectional RNN, resulting in a model consisting of a bidirectional RNN and a pooling layer (see figure 3.8). In their model, a recurrent structure is applied to capture the contextual information when learning word representations. In this way, it

may introduce considerably less noise compared to traditional window-based neural networks. Also, they capture the key components in texts by employing a max-pooling layer that automatically judges which words play key roles in text classification.

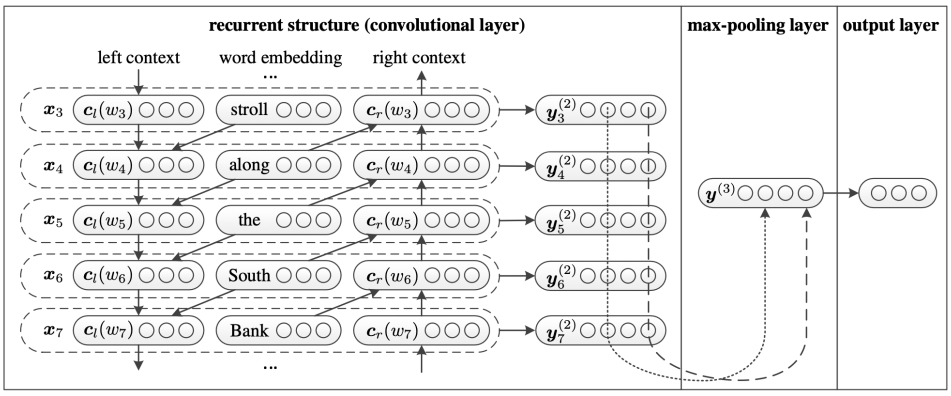


Figure 3.8: The overall structure of RCNN (Lai et al. [2015]). This figure is a partial example of the sentence "A sunset stroll along the South Bank affords an array of stunning vantage points", and the subscript denotes the position of the corresponding word in the original sentence.

The first part of the model is the recurrent structure. In this network, the hidden state of the i-th position is affected by the context at both the left and right sides:

$$c_l(w_i) = f(W^l c_l(w_{i-1})) + W^{sl} e(w_{i-1})$$

$$c_r(w_i) = f(W^l c_r(w_{i+1})) + W^{sr} e(w_{i+1})$$

where w_{i-1}, w_{i+1}, $e(w_{i-1})$, $e(w_{i+1})$ denote the hidden state and word embeddings at position $i-1$ and $i+1$, respectively.

In Zhou et al. [2020], they report their implementation of the TextRCNN encoder on several public datasets and compared it with the new method (HiAGM) they proposed. Tencent implemented the TextRCNN model Liu et al. [2019] reached 0.8313 of Micro-F1 on the RCV1 dataset. This modelreachesh the best score (Micro-F1: 0.5526) on the O*NET training subset.

3.3.2 Recursive Regularization

This method is initially proposed by Gopal and Yang [2013]. This method incorporates the hierarchical dependencies between the class labels into the regularization structure of the parameters thereby encouraging classes nearby in the hierarchy to share similar model parameters. Intuitively, it is based on the assumption that the

nearby classes in the hierarchy are semantically close to each other and hence share similar model parameters.

The Structural Risk Minimization framework prescribes choosing f to minimize a combination of the Empirical Risk based on the training dataset and a regularization term to penalize the complexity of f. Typically the prediction function f is parameterized by an unknown set of parameters w which are then estimated in the learning process. The estimated parameters $\hat{w}$ is given by:

$$\hat{w} = arg\min_{w} \lambda(w) + C \times R_{emp}$$

where R_{emp} denotes the Empirical Risk or Loss on the training dataset, $\lambda(w)$ denotes the regularization term and C is a parameter that controls the trade-off between fitting to the given training instances and the complexity of f.

The most innovative point is the regularization form they introduced:

$$\lambda(W) = \sum_{n \in N} \frac{1}{2} ||w_n - w_{\pi(n)}||^2$$

The idea of introducing regularization terms works very well because it introduce hierarchical dependencies between the class labels into the regularization structure, and therefore determine similarity and shared parameters.

This work inspired a lot of following studies for hierarchical text classification, some models incorporate this approach directly, like HiAGM.

3.3.3 HMCN

Hierarchical **M**ulti-Label **C**lassification **N**etworks is proposed by Wehrmann et al. in 2018. This network is designed for HMC problem specifically and reach state-of-the-art results on HMC datasets, e.g.: CELLCYCLE, EISEN, GASH and so on. The most innovative point of HMCN is designed to optimize local and global loss functions simultaneously while penalizing hierarchical violations. For each hierarchical level, there is one local output and a local loss function for backpropagating the gradients from classes at the specific level. For the entire network, the global output captures the cumulative information and the function backpropagates the gradients from all classes over the whole hierarchy.

According to whether there is a recurrent architecture, they proposed two variants of HMCN: a feed-forward HMCN-F and a recurrent architecture HMCN-R.

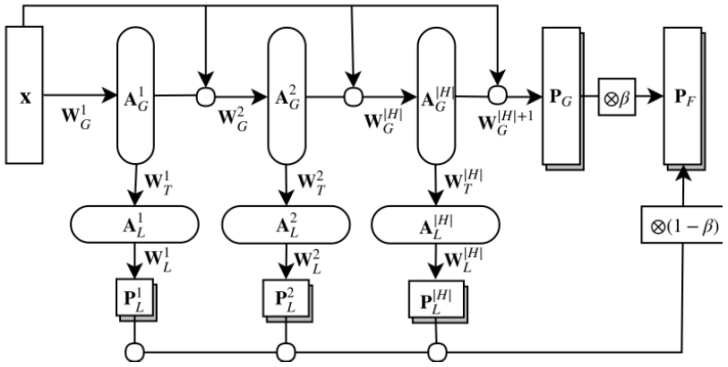


Figure 3.9: HMCN-F architecture (Wehrmann et al. [2018]).

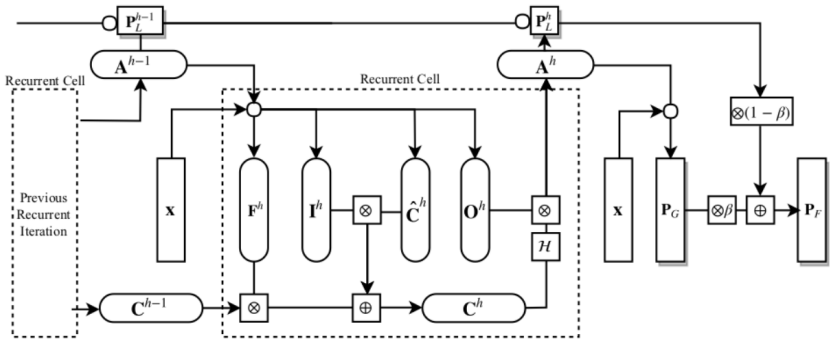


Figure 3.10: HMCN-R architecture (Wehrmann et al. [2018]).

1. **HMCN-F** As shown in Figure 3.9, there are two main data flows in HMCN-F.

   - The global flow. It starts from the input layer x and traverses all fully-connected (FC) middle layers A_G^i, and reaches the output layer P_G.
   - The local flows. Starting from the inputs layer x, then pass by corresponding middle layers A_G^i and the specific local FC layers A_L^i. It ends at the local output layers P_L^i.

   In the final step, all the local outputs are concatenated and pooled with global output to generate a consensual prediction as the final output.

2. **HMCN-R** Inspired by LSTM networks, Wehrmann et al. proposed the recurrnt version of HMCN). In HMCN-R, the output of each iteration from the unrolling recurrent network concerns a hierarchical level. As shown in Figure 3.10, the architecture of HMCN-R is quite similar to the original LSTM networks. For the h^{th} level, the output A^h is computed by C^h and output gate O^h, where C^h denotes the cell state and is modified by forget gate F^h and

input gate I^h.

Compared to HMCN-F, whose amount of parameters grows with the hierarchical levels increase, HMCN-R keeps the number unchanged even for very deep hierarchies.

Loss Function plays an important role in HMCN. It minimize the sum of local and global loss as following:

$$\min_W(\mathcal{L}_L + \mathcal{L}_G + \mathcal{L}_H)$$

These are how the local and global loss computed:

$$\mathcal{L}_L = \sum_{h=1}^{|H|}[\varepsilon(P_L^h, Y_L^h)]$$

$$\mathcal{L}_G = \varepsilon(P_G, Y_G)$$

where $\varepsilon(\hat{Y}, Y)$ denotes the binary cross entropy, which has to be minimized:

$$\varepsilon(\hat{Y}, Y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{|C|}[Y_{ij} \times \log(\hat{Y}_{ij}) + (1 - Y_{ij}) \times \log(1 - \hat{Y}_{ij})]$$

$\mathcal{L}_H$ indicates the *hierarchical violation loss*, which happens when the generated prediction score of a child node Y_{in} is larger than the score of its parent node Y_{ip}. It is calculated as:

$$\mathcal{L}_{H_i} = \lambda \max\{0, Y_{in} - Y_{ip}\}^2$$

In this formula, $\lambda \in \mathbb{R}$ is employed for regulating the import of the penalty for the hierarchical violations in the final loss function. If the λ is too large, the result may be biased towards predicting smaller values within deeper layers. However, if the λ is too small, the network may result easier in inconsistent paths which means it tends to ignore the hierarchical structure compared to the statistical characteristics.

Among most of the datsets they experiment on, HMCN-F always report better results compared with HMCN-R ($\lambda = 0.1$). In our experiment, we applied the implementation of HMCN-F from Tencent [3]. During all the training process on the O*NET dataset, the value of λ is set to 1e-5.

[3] `https://github.com/Jinqiao-Li/NeuralNLP-NeuralClassifier/tree/master`

3.3.4 HiAGM

Hierarchy-**A**ware **G**lobal **M**odel (HiAGM) combines a traditional text classification model with a hierarchy encoder. With the predefined hierarchy and corpus, HiAGM leverages the prior hierarchy information based on Bayesian statistical theory. (Zhou et al. [2020])

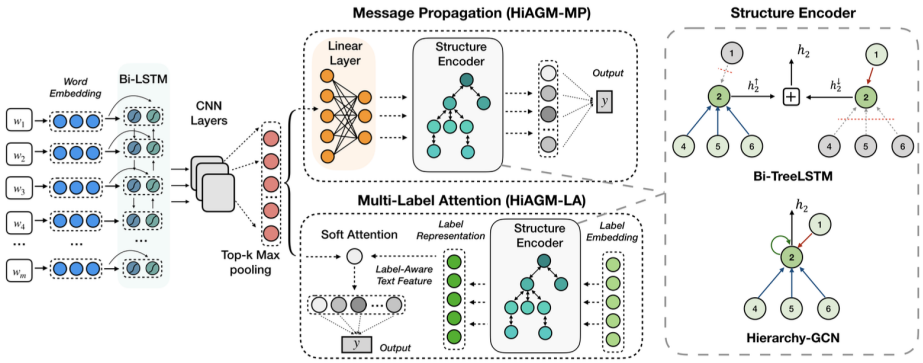


Figure 3.11: The overall structure of HiAGM (Zhou et al. [2020]).

As depicted in Figure 3.11, HiAGM models adopt two types of structure encoders with prior hierarchy information to aggregate node information: Tree-LSTM and graph convolutional neural networks (GCN). For the top-down dataflow, encoders employ the prior hierarchy information as $f_c(e_{i,j}) = \frac{N_j}{N_i}$, and $f_p(e_{i,j}) = 1.0$ for the bottom-up direction.

1. **Bidirectional Tree-LSTM**

   HiAGM models employ the Tree-LSTM similar to syntax encoders (Tai et al. [2015]). It allows the mini-batch training for the recursive computational module, because the predefined hierarchy is identical to all samples. The transformation between nodes is as follows:

$$i_k = \sigma(W_{(i)}v_k + U_{(i)}\widetilde{h}_k + b_{(i)}),$$

$$f_{k,j} = \sigma(W_{(f)}v_k + U_{(f)}h_k + b_{(f)}),$$

$$o_k = \sigma(W_{(o)}v_k + U_{(o)}\widetilde{h}_k + b_{(o)}),$$

$$u_k = \tanh(W_{(u)}v_k + U_{(u)}\widetilde{h}_k + b_{(u)}),$$

$$c_k = i_k \odot u_k + \sum_j f_{k,j} \odot c_j,$$

$$h_k = o_k \odot \tanh(c_k),$$

where h_k represent the hidden state and c_k refers to the memory cell state of node k.

Also, the HiAGM implement the bidirectional Tree-LSTM to induce label correlations with the following formulas:

$$\widetilde{h}_k^{\uparrow} = \sum_{j \in child(k)} f_p(e_{k,j}) \widetilde{h}_j^{\uparrow},$$

$$\widetilde{h}_k^{\downarrow} = f_c(e_{k,p}) \widetilde{h}_p^{\downarrow},$$

$$h_k^{bi} = h_k^{\uparrow} \oplus h_k^{\downarrow},$$

where $h_k^{\uparrow}$ and $h_k^{\downarrow}$ represent the bottom-up and top-down manners respectively, and h_k is calculated as $TreeLSTM(\widetilde{h}_k)$. h_k^{bi} represent the final hidden state of node k and $\oplus$ indicates the hidden states concatenation computaion.

2. **Hierarchy-GCN**

   HiAGM models introduce the hierarchy-GCN to represent the prior hierarchy information. The GCN integrates the text semantics with prior hierarchy information. In the Hiearchy-GCN, edges contains three hierarchy path direction: top-down, bottom-up, and self-loop. Each edge represents a pair-wise label correlation feature (as dipicted in Figure 3.11). In order to avoid over-parameterized edge-wise weight matrixes, a weighted adjacent matrix is used to simplify the transformation process.

   The hidden state of node k is encoded as: $N(k) = \{n_k, child(k), parent(k)\}$.

$$u_{k,j} = a_{k,j} v_j + b_l^k,$$

$$g_{k,j} = \sigma(W_g^{d(j,k)} v_k + b_g^k),$$

$$h_k = ReLU(\sum_{j \in N(k)} g_{k,j} \odot u_{k,j}),$$

   where $W_g^{d(j,k)} \in \mathbb{R}^d$, $b_l \in \mathbb{R}^{N \times d}$, and $b_g \in \mathbb{R}^N$. $a_{k,j}$ represent the hierarchy probability from node k to node j: $f_{d(k,j)}(e_kj)$. According to the Bayesian theory, the top-down direction employs $f_c(e_{j,k}) = \frac{N_k}{N_j}$, and bottom-up edges use $f_p(e_{j,k}) = 1$, and the self-loop is $a_{k,k} = 1$. Also, $d(j,k)$ denotes the three hierarchical directions from node j to node k.

Also, as shown in Figure 3.11, there are two variants of HiAGM for hybrid information aggregation:

1. **Hierarch-Aware Multi-Label Attention (HiAGM-LA)** This is a multi-label attention model. It updates label representations with the structure encoder and generates label-aware text features with multi-label attention mechanism. The calculation of attention value α_{kj} is as follows:

$$\alpha_{kj} = \frac{e^{s_j h_k^T}}{\sum_{j=1}^{n} e^{s_j h_k^T}}, v_k = \sum_{i=1}^{n} \alpha_{ki} s_i$$

   where α_{kj} denotes the attention of the $i-th$ text feature vector for the $k-th$ label. $S \in \mathbb{R}^{n \times d_c}$ indicates the text representation. The inductive label-aligned text features $V \in \mathbb{R}^{C \times d_c}$ is calculated based on the α_{kj}.

2. **Hierarchical text feature propagation (HiAGM-TP)** Another is a text feature propagation model. It propagates text representations throughout the holistic hierarchy, so that it can get label-wise text features with the fusion of label correlations. With a single linear transformation, the node inputs V is calculated as follows:

$$V = MS$$

   where $V \in \mathbb{R}^{C \times d_v}$ indicates the node inputs, $M \in \mathbb{R}^{(n \times d_c) \times (C \times d_v)}$ is the trainable weight matrix, $S \in \mathbb{R}^{n \times d_c}$ is the text features.

In this work, I run both variants combined with different encoders (Tree-LSTM and GCN)on the O*NET sub-training set.

4 Experiment

In our experiments, I first carry out a series of experiments with transformer-based flat classification models on O*NET training data to find the best way of addressing the language gap problem in this task. In addition, these flat classification models serve as the baseline for evaluating the performance of hierarchical models, expecting to demonstrate the effectiveness of global hierarchical models. These hierarchical models report good results on the public datasets mentioned in section 2.1 and were state-of-the-art models when it was proposed.

We selected the two best-performing models from baseline and hierarchical experiments separately to classify the job ads data. These predictions are evaluated manually as our ground truth.

As some hierarchical models (like HiAGM) require too much GPU memory during the encoding and training process, I cannot experiment with these models successfully on our hard device with the whole O*NET ontology hierarchy. Considering the hardware limitation, I create subsets from the training data set to compare all the models' performance directly. Figure 4.1 gives an overview of our experiments.

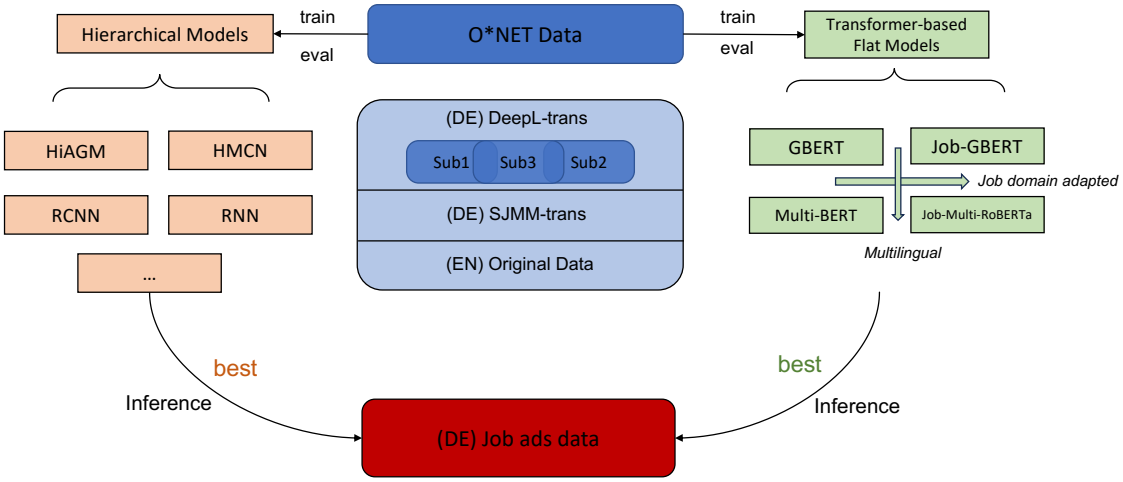


Figure 4.1: Experimental pipeline.

4.1 Baseline Experiments

Flat classification models were trained on each hierarchical level as the baseline for later comparison with hierarchical classification models. In addition, at this phase, we compared two methods to address the language gap between O*NET ontology data and job advertisements. 1) One method is to employ German transformer-based models, like the German BERT model (for convenience, I call it 'gbert' in the following report). These models are trained with German o*NET data that is translated by two machine translation engines: SJMM and DeepL. Translation qualities of these two versions of German O*NET data are also compared. 2) The second method to address the language gap is to apply multilingual transformer-based models, like XML-RoBERTa. These multilingual models are trained on the original English O*NET data and then evaluated on the German test set.

Furthermore, inspired by Gnehm et al.'s previous work, which proved the effectiveness of job-ads domain adaption for transformer-based models, I also experiment with transformer-based models trained with in-domain job ads data. As a result, both GBERT and Multi-BERT models can be compared with job ads adapted models respectively.

Overall, as depicted in Figure4.1, there are four transformer-based flat models trained with three versions of O*NET data in this series of experiments.

In the training process, all these experiments are only conducted on the highest GWA level for simplicity. Then, we run experiments on the IWA and DWA labels only with the best configuration from gained GWA experience. After that, we selected the best three models from different levels to create predictions on job ads separately.

4.1.1 Experimental Data

4.1.1.1 O*NET Data

As mentioned in Section 3.2.3, the original ontology English data is translated to German using the SJMM system and DeepL API. The baseline models are also trained with these two translated German ontology data. In this way, we could examine how the quality of translation affects the classification results.

We divided the data into training, validation and test sets in a ratio of 7:1:2 (see table 4.1). Each splitted data set contains the original English, sjmm-translated German

and DeepL-translated German data of text and label. Table 4.2 gives details about the three different version of O*NET text data.

Set	Train	Validation	Test
Size	16480	2354	4708

Table 4.1: Overview of O*NET data sets.

Text	Label	Language	Avg_len(word)	Note
Task	GWA Title	EN	13.82	original ontology text
Task_de	GWA_de	DE	14.20	translated by SJMM system
Task_deepl_de	GWA_deepl_de	DE	14.90	translated by DeepL API

Table 4.2: Training data on the GWA level. *Label* only denotes its name on the GWA hierarchial level. *Avg_len* is the average length of text among all samples.

4.1.1.2 Job Ads Data

As shown in the diagram 4.1, the best-performing model from baseline and hierarchical models are selected to do inference on job ads data.

As the baseline models are trained separately on different classification levels, we predicted the top5 best classes for each sample at three levels. The last layer of baseline models is *softmax*, so the model also gives a probability score for each prediction. The five predictions are listed in descending order of probability.

For 200 samples on three hierarchical levels, the top three predictions are noted by one annotator, who is a German-speaking domain expert. Each predicted label is notes as one of: {1: 'totally right', 0.5: 'acceptable', 0: 'wrong'}.

Table 4.3 gives an example of what the inferred job ads data looks like.

4.1.2 Training Models

In this work, for better comparison, four models pre-trained with different methods were selected and fine-tuned with ontology data. Overall, these models were trained based on transformer: BertForSequenceClassification and XLMRobertaForSequenceClassification. These two transformers add a sequence classification head on top of BERT Model (Devlin et al. [2019]) and XLM-RoBERTa Model (Conneau et al. [2020]) respectively.

Text	Level	Predicted label	Score	Eval
Verpackungsarbeiten allgemein und in Medizinaltechnik	GWA	Handhabung und Bewegung von Objekten	0.6834	1
		Ausführen allgemeiner körperlicher Aktivitäten	0.1804	0.5
		Reparatur und Wartung von mechanischen Geräten	0.0376	0
	IWA	Bedienung von industriellen Verarbeitungs- oder Produktionsanlagen.	0.1911	0.5
		Objekte verpacken.	0.1189	1
		Industrielle Materialien für die Verarbeitung oder Verwendung vorbereiten.	0.0686	0.5
	DWA	Führen eines Inventars an medizinischem Material und Ausrüstung.	0.0847	0
		Medikamente oder medizinische Lösungen zubereiten.	0.0346	0
		Entfernen von Produkten oder Werkstücken aus Produktionsanlagen.	0.0231	0

Table 4.3: Details about job ads data evaluation. *Eval* is the human annotation.

Before putting text into transformers, I encoded the inputs with tokenizer which split it into a series of tokens for later training. Considering the tasks' length distribution (see figure: 3.6), I set the max length as 64.

I applied Pytorch Lightening [1]for training models, which is a lightweight Pytorch library and provides structured modules for model, data loader, etc. In this way, I am able to build the transformer-based code in a very clean and efficient way.

In the forwarding process, as this is a classification problem, I calculated the cross entropy loss of unnormalized logits value in each step.

When training these baseline models for flatten classification, I use the AdamW optimizer and learning rate of 2e-5. The maximum number of 10 epochs set to 10 with early stopping strategy and the batch size is 16.

During the training process, I also applied the early stop strategy which is integrated into the callback class in the Pytorch lightening structure. Validation loss is set as monitor, minimum delta as 0.01, and patience as 2, which means the training process may stop early if the validation loss has not improved more than 0.01 in two continuous epochs.

1. **GBERT** Considering the job ads data and translated ontology data are both

[1]`https://www.pytorchlightning.ai`

in German, we select the transformer-based German model which was published by Chan et al. [2020] in 2020. This language model was trained collaboratively by the makers of the original German BERT and the dbmdz BERT and shown as the best German model at this time. In the experiments, I imported the pre-trained model from HuggingFace [2] and finetuned it with our own tasks data. In this report, this model is called 'GBERT' for simplicity.

2. **Job-GBERT** In Gnehm et al. [2022b], they applied transfer learning and domain adaptation based on the GBERT with job ads corpora with the masked language modeling task. They proved the efficiency of domain-adaptation of language models via continued pre-training and vocabulary customization. In this experiment, I also employed the Job-GBERT model [3] they released, and expect an improvement for this job classification task based on previous GBERT model.

3. **Multi-BERT** As this work involves two different languages (English & German), I also used a multilingual model[4] from HuggingFace to see if it has any help in later evaluation on job ads data. This BERT model is pre-trained on the top 104 languages with the largest Wikipedia using a masked language modeling (MLM) objective.

4. **Job-Multi-RoBERTa** The Multilingual job model is shared by Gnehm et al. in 2022. They fine-tuned the XLM-RoBERTa Model transformer (Conneau et al. [2020]) with the job in-domain data. Similarly to the first two models, we expect the Job-Multi-RoBERTa model outperforms the Multi-BERT model by getting improvement from domain-adaption.

4.2 Hierarchical Classification Experiments

Besides the first series of experiments about flat classification, I also designed experiments with hierarchical models. As these models are designed to capture the hierarchical relations, they are expected to have better performance on this hierarchical text classification task.

In addition, according to the baseline experiments, all models perform better on DeepL engine translated German data. Therefore, we only conduct further hierarchical experiments on the DeepL translated data.

[2] `https://huggingface.co/deepset/gbert-base`

[3] `https://huggingface.co/agne/jobGBERT`

[4] `https://huggingface.co/bert-base-multilingual-cased`

4.2.1 Data Preprocessing

After careful examination of DeepL translated German data, we find it is kind of noisy. During the translation process, some irrelevant characters are added. Also, there are some German stop words in the *Task* text which do not help models understand the text. Therefore, data is pre-processed before experimenting with hierarchical models.

Overall, there are two steps in the data preprocessing pipeline: data cleaning and creating input data format. O*NET training data pass through the two steps to fit with hierarchical classification models. While the job ads data only goes through the data cleaning step and the inference are made based on the cleaned data.

For all the hierarchical classification models, we used the same pre-processed O*NET training data. However, there are two types of input formats to fit different models' requirements.

4.2.1.1 Data Cleaning

As mentioned before, according to baseline modes' results, I only clean the DeepL translated German data at this phase.

For the text of *Task*, I removed both the special characters and stop words. While for the labels, I only removed the special characters. This is because the labels are usually shorter phrases rather than full sentences and there are few stopping words. Besides, in this task, labels are transformed into numerical representations instead of treated as a sequence.

- **Special characters** The translated data is kind of noisy. For example, there are periods at the end of all the labels at the IWA and DWA levels, while none for the GWA level. In order to compare the translation qualities, DeepL translator adds some special characters to keep consistency with previous SJMM translated text. Therefore, I removed these special characters to keep the data clean and format uniform before experimenting with hierarchical models. Table 4.4 gives an example of what a DWA label looks like after pre-processing.
- **Stop words** Following the operation that is carried out by all the hierarchical classification models (like HMCN, HiAGM), I remove the stop words from all the train, validation and test sets. I used the NLTK (Natural Language Toolkit) [5] package and its inner German stop words list which contains 232

[5] `https://www.nltk.org/`

most common words, including *aber*, *denn*, *muss*, *zwar*, etc.

DeepL translated DWA	Preprocessed DWA
['Analyse der Auswirkungen von rechtlichen oder regulatorischen Änderungen.']	Analyse der Auswirkungen von rechtlichen oder regulatorischen Änderungen

Table 4.4: An example of data cleaning.

4.2.1.2 Input Data Format

I create the *train.json*, *val.json* and *test.json* files for each hierarchical model. Overall, there are two input formats for all the hierarchical models whicle training with O*NET data. Table 4.5 gives an example of the first input format. *doc_token* is the list of tokens of cleaned Tasks and all the German stop words are removed. *doc_label* is the list of all labels assigned to the task. In order to make the hierarchical relations more clear, the *doc_label* is organized as ['GWA', 'GWA–IWA', 'GWA–IWA–DWA']. For samples which are assigned to more than one hierarchical label path, it is organized like: ['GWA1', 'GWA1–IWA1', 'GWA1–IWA1–DWA1', GWA1–IWA2', 'GWA1–IWA2–DWA2']

However, in the training set, there are a few samples that have identical IWA and DWA labels. In the pre-processing phase, I delete the DWA labels that are the same as the IWA labels. As a result, there are only two levels for these samples: ['GWA', 'GWA–IWA'].

{
"doc_token": ["Einsatz", "Geräten", "Gabelstaplern", "automatischen", "Zügen", "Transport", "Postcontainern"],
"doc_label": ["Betrieb von Fahrzeugen mechanisierten Geräten oder Ausrüstungen",
"Betrieb von Fahrzeugen mechanisierten Geräten oder Ausrüstungen–Bedienen von Transportmitteln oder Fahrzeugen",
"Betrieb von Fahrzeugen mechanisierten Geräten oder Ausrüstungen–Bedienen von Transportmitteln oder Fahrzeugen–Bedienung von Fahrzeugen oder Materialtransportgeräten"]
}

Table 4.5: A pre-processed sample for input format1. This format is used for TextRNN, TextRCNN with recursive regularization. It also serves the HMCN model without recursive regularization.

Table 4.6 gives an example of another input format. This format mainly serves for hierarchical models which are trained without recursive regularization. Generally, it is organized as: ['GWA', 'IWA', 'DWA']. In this example, as it is assigned to more than one hierarchical path, it ends more than the normal three labels.

{
“doc_token”: [“Überprüfen”, “Sie”, “endgültigen”, “Entwürfe”, “schlagen”, “Sie”, “Bedarf”, “Verbesserungen”],
“doc_label”: [“Informationen erhalten”, “Studieren Sie Details von künstlerischen Produktionen”, “Überprüfen Sie Kunst oder Designmaterialien”, “Kommunikation mit Vorgesetzten Kollegen oder Untergebenen”, “Kommunizieren Sie mit anderen über Spezifikationen oder Projektdetails”, “Arbeiten Sie mit anderen zusammen um Entwürfe zu entwickeln oder zu verfeinern”]
}

Table 4.6: A pre-processed sample for input format2. This format is used for TextRNN, TextRCNN without recursive regularization, and the HiAGM model.

4.2.1.3 Creating Taxonomy Structure

Like the implementation in the previous studies about hierarchical classification, I need to create a file describing the hierarchical structure before starting training models. The file serves for all the hierarchical models we tested in this project and it is called *‘onet_de.taxonomy’*. Tablel 4.7 gives an example of the taxonomy structure of a small part of the DeepL translated German O*NET sub3. We follow this structure to create the taxonomy files for the whole and subset of O*NET data separately.

In Appendix, table 1 gives part of the ontology structure of the RCV1 dataset. Labels in this dataset are more simple and easier for understanding.

4.2.2 Sub-Dataset

The O*NET ontology has a much larger and more complex hierarchy than the public dataset we talked about in Section 2.1. As a result, some global hierarchical models, such as HiAGM, cannot be directly applied to the entire training set, as their GPU memory requirements exceed the hardware availability we have at hand. Since I want to compare different models as a proof of the concept, I perform a vertical slice of the original dataset and extracted three subsets for training.

Three different subsets are extracted from O*NET ontology data. For each subset, I extracted five GWA labels and then find all the samples from the train, validation and test sets that are assigned to any of the GWA labels.

Sub1 contains the five most frequent GWA labels and their child labels in IWA and DWA levels. Sub2 is composed of the five least frequent GWA labels and their child

Parent Node	Level	Children Nodes
Root	GWA	Konflikte lösen und mit anderen verhandeln
		Verkaufen oder Beeinflussen anderer
Konflikte lösen und mit anderen verhandeln	IWA	Verträge oder Vereinbarungen aushandeln
		Lösen Sie personelle oder betriebliche Probleme Schlichtung von Streitigkeiten
Verträge oder Vereinbarungen aushandeln	DWA	Aushandeln von Verträgen für Transport Vertriebs oder
		Logistikdienstleistungen Aushandeln von Kauf oder Leasingverträgen für Produkte oder Dienstleistungen
Verkaufen oder Beeinflussen anderer	IWA	Werbung für Produkte Dienstleistungen oder Programme
		Verkaufen Sie Produkte oder Dienstleistungen
Werbung für Produkte Dienstleistungen oder Programme	DWA	Produkte Dienstleistungen oder Veranstaltungen vermarkten
		Durchführung von Marketingaktivitäten
Verkaufen Sie Produkte oder Dienstleistungen	DWA	Vermarktung von Gesundheitsprodukten oder dienstleistungen
		Anpassung von Finanzprodukten oder dienstleistungen an die Bedürfnisse der Kunden

Table 4.7: Sample taxonomy structure of sub-O*NET dataset. *Level* refers to *Children Nodes'* hierarchical level in the O*NET structure.

labels in two lower hierarchical levels. And then I randomly generated 5 numbers between 0 to 37, which are [0, 10, 15, 26, 33]. Using these numbers as the index, I select the corresponding GWA labels to form the sub3 which are sorted in a descent order by frequency. Table 4.8 shows the size of these three subsets. It includes number of labels at each hierarchical level and also the number of samples in each data set. The details about these three subsets and their corresponding appearance times are in Table 4.9.

As **sub3** is selected randomly, it can represent the label imbalance distribution in the whole O*NET ontology set better. Therefore, sub3 is used as a generic dataset and compared the performance of all models on it (see results in 5.8). In addition, in order to find out global hierarchical models' performance in more extreme situations, I experiment HiAGM-LA model on three datasets that contain labels with different frequencies and compared their performance.

Subset	GWA	IWA	DWA	Train	Val	Test
sub1	5	123	765	6295	943	1806
sub2	5	10	51	324	44	96
sub3	5	70	450	3412	512	971

Table 4.8: O*NET subsets statistics. *GWA*, *IWA* and *DWA* means the number of classes at each level. *Train*, *Val*, *Test* corresponding to the samples in each set.

Subset	GWA Label	Num(Train)	Num(Val)	Num(Test)
sub1	Handhabung und Bewegung von Objekten	2120	319	616
	Dokumentieren Aufzeichnen von Informationen	1217	196	345
	Führung Leitung und Motivation von Untergebenen	1002	156	287
	Kreativ denken	978	150	279
	Ausführen allgemeiner körperlicher Aktivitäten	978	122	279
sub2	Besetzung der Organisationseinheiten	90	12	19
	Konflikte lösen und mit anderen verhandeln	89	17	21
	Identifizierung von Objekten Aktionen und Ereignissen	75	6	32
	Coaching und Entwicklung anderer	40	5	14
	Aufbau und Pflege von zwischenmenschlichen Beziehungen	30	4	10
sub3	Handhabung und Bewegung von Objekten	2120	319	616
	Überwachung von Prozessen Materialien oder Umgebungsbedingungen	548	89	163
	Andere ausbilden und unterrichten	484	72	131
	Verkaufen oder Beeinflussen anderer	171	15	40
	Konflikte lösen und mit anderen verhandeln	89	17	21

Table 4.9: Details in three subsets. *Num(Train)*, *Num(Val)*, *Num(Test)* are the numbers the corresponding GWA label appeared in the train, validation and test sets.

4.2.3 Implementation Details

I mainly employed the NeuralClassifier Tookit[6] (Liu et al. [2019]) for implementing these models. This is an open-source hierarchical multi-label classifier toolkit provided by Tencent.

Overall, on the whole O*NET training dataset, I build three following classifiers:

1. A hierarchical classifier with HMCN. HMCN model calculate the training loss by combining global and local loss as indicated in the work Wehrmann et al.

[6] `https://github.com/Tencent/NeuralNLP-NeuralClassifier`

[2018]. We used the implementation by Tencent directly and set the parameters according to the size of O*NET training set.

2. A hierarchical classifier using hierarchical penalty. This kind of model is called as *hierar* model for simplicity. Many models can be used for this classifier: TextCNN, TextRNN, TextRCNN, AttentiveConvNet, Transformer and so on. In this project, I mainly tried the TextRNN and TextRCNN models. In the previous study (Liu et al. [2019]), TextRCNN model report best results on the RCV1 data sets. Compared to the non-hierarchical classifier, this classifier applied hierarchical penalty (set to 1e-5) and recursive regularization while calculating loss during training process.

3. A non-hierarchical classifier. In the later results tables, I call it as *flat* model for simplicity. This kind of classifiers are employed to evaluate the effect of hierarchical penalty, by comparing with the hierarchical models combined with recursive regularization.

Furthermore, I also run the HiAGM with two different variants (HiAGM-LA and HiAGM-TP) and two encoding methods (Tree-LSTM and GCN) on sub-O*NET ontology data to compare their performance. Overall, three different configurations of HiAGM: HiAGM-LA_treela, HiAGM-LA_gcn, HiAGM-TP_treela and HiAGM-TP_gcn.

For hierarchical multi-label classifiers, I use *BCELoss* as the loss function and add a recursive regularization. With this regularization framework, the model encourages the nearby hierarchical classes to share similar parameters. In addition, using such a regularization, is more suitable for large-scale hierarchical multi-label classification tasks, like O*NET training data. In the training process, I set the maximum number of epochs as 100, and the batch size is 64. According to the *tasks* length distribution analysis, I set the maximum length of encoding to 64.

4.3 Evaluation Metrics

To evaluate the quality of the model's classification, I applied some metrics, including precision, recall, accuracy, f1 score, and hamming loss which is specifically for multi-label classification problems. Same as many previous studies, we mainly refer to the micro-F1 score to decide the best model in the same configurations.

The four components of confusion matrix are used to compute these metrics:

- True Positives (TP): Number of samples correctly predicted as "positive."

- False Positives (FP): Number of samples wrongly predicted as "positive."
- True Negatives (TN): Number of samples correctly predicted as "negative."
- False Negatives (FN): Number of samples wrongly predicted as "negative."

4.3.1 F1 Score

The goal of the F1 score is to combine the precision and recall metrics into a single metric. The F1 score is defined as the harmonic mean of precision and recall. At the same time, the F1 score also work well on imbalanced data.

The calculations for precision and recall:

$$Precision = \frac{TP}{TP + FP} \quad , \quad Recall = \frac{TP}{TP + FN}$$

So, the formula for F1 Score is:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

F1-score range from 0 to 1, where 0 is the worst value and the best is 1.

1. Macro-averaged F1 Score
   The macro-averaged F1 score of a model is just a simple average of the class-wise F1 scores obtained.

   $$Macro\ F1 = \frac{\sum_{i=1}^{n} F1\ Score_i}{n}$$

   where n is the number of classes in the data set.

2. Micro-averaged F1 Score
   Micro-averaging essentially computes the proportion of correctly classified samples out of all samples in the data set. Actually, this definition is equal to what is used to calculate overall accuracy and precision with micro strategy.

   Compared to other computing strategies, micro-averaging gives each sample the same importance. And it is also the preferred behavior to assess the quality of multi-label classification problems.

   In this project, I mainly referred to the micro F1-score while evaluating the models' performance. In this whole experiment, while I talk about the 'best'

model, it always means the model that gets the highest micro-F1 score in the configurations.

3. Support-weighted F1 score
   This weighted strategy is ideal for computing the F1 score of imbalanced data. It computes class-wise F1 score, and the weights are based on the number of samples in each class.

$$Weighted\ F1\ Score = \sum_{i=1}^{n} w_i \times F1\ Score_i$$

   where n is the number of classes, w_i is the ratio of the number of samples in class i in the total number of samples.

   Three of the computation strategies are all applied and compared while evaluating the baseline models on the IWA and DWA level (see Table 5.2).

4.3.2 Hamming Loss

The Hamming loss is the fraction of labels that are incorrectly predicted. The loss value range from 0 to 1, where 1 is the worst value and 0 is the best value.

In multiclass classification, the Hamming loss corresponds to the Hamming distance between y_true and y_pred which is similar to the Zero one loss function. However, the Hamming loss penalizes individual labels, instead of penalizing predictions that do not strictly match true sets in zero-one loss.

Computing the average Hamming distance in pytorch [7] for multiclass tasks:

$$Hamming\ loss = \frac{1}{N \cdot L} \sum_{i}^{N} \sum_{l}^{L} 1(y_{il} \neq \hat{y}_{il})$$

Where y is a tensor of target values, $\hat{y}$ is a tensor of predictions, and $\bullet\_il$ refers to the l-th label of the i-th sample of that tensor.

[7] `https://torchmetrics.readthedocs.io/en/stable/classification/hamming_distance.html`

4.3.3 Average Precision Score

In evaluating the model's predictions for the job advertisement data, I also used the average precision (AP) metric from the scikit-learn package. In the actual calculations, I obtained two versions of the average accuracy figures: hard and soft, i.e., considering "half correct" as "wrong" and "totally correct", respectively. Here, "half correct", "wrong" and "totally correct" refer to the human annotations discussed in the section 4.1.1.2".

Average precision summarizes the precision-recall curve as a weighted average of the precision achieved at each threshold, with the increase in recall over the previous threshold as the weight:

$$AP = \sum_{n} (R_n - R - n - 1) P_n$$

where P_n denotes the precision and R_n denotes the recall.

5 Results

In this Chapter, I am going to analyze all models' performance on the O*NET test set and Job Ads set. First, I compared four baseline models' performance on the GWA hierarchical level of the O*NET dataset and discussed how that influenced the choices of training data and the best baseline model for the IWA and DWA levels. Besides, I used sample data on the GWA level to show how is the translation quality effect prediction results. Second, I analyzed hierarchical models' performance on the O*NET set. In order to compare well with baseline models, I also create an O*NET subset where I experiment with all models on it. Third, I analyze the trained models' predictions on the Job Ads set which is also the essential goal of this project. In the end, based on the previous results, I analyze the difficult points of this project, especially the O*NET ontology hierarchy.

5.1 Baseline Results

Model	Trainging Data	Micro-F1	Macro-F1	Recall	Precision	Hamming_loss
GBERT	task_en	0.5291	0.3780	0.4400	0.3827	0.4709
Job-GBERT		0.5138	0.3719	0.4373	0.3901	0.4862
Multi-BERT		0.5070	0.3417	0.3989	0.3429	0.4930
Multi-Job-RoBRETa		0.5340	0.3852	0.4374	0.3799	0.4660
GBERT	task_sjmm_de	0.5458	0.4751	0.4803	0.4888	0.4542
Job-GBERT		0.533	0.4535	0.4507	0.4879	0.467
Multi-BERT		0.5317	0.4371	0.4347	0.4845	0.4683
Multi-Job-RoBRETa		0.5453	0.4549	0.4524	0.488	0.4547
GBERT	task_deepl_de	0.5804	0.5037	0.5089	0.5074	0.4196
Job-GBERT		**0.5819**	**0.5068**	**0.5103**	**0.5208**	**0.4181**
Multi-BERT		0.5487	0.4704	0.4801	0.481	0.4513
Multi-Job-RoBRETa		0.5763	0.4862	0.4866	0.5057	0.4237

Table 5.1: Models' scores on the GWA level with different training data. *'task_en'* means the original text in English. *'task_sjmm_de'* means German text translated by SJMM system. *'task_deepl_de'* means German text translated by DeepL API. *Recall* and *Precision* are computed with 'macro' strategy.

Table 5.1 gives the performance of the baseline model at the GWA level, including

the raw English data and the German data translated by SJMM and DeepL. It reports the F1 scores calculated with the micro and macro strategies, as well as the recall and precision scores calculated with the macro strategy. For more information, Table 2 and Table 3 in the Appendix give all metrics calculated with the three average strategies on the two versions of the German data.

Looking at the results of the three versions of the O*NET data, we can see that the comparison of all metrics is relatively consistent across the four models. In general $task/_{e}n < task/sjmm/de < task/_{d}eepl/de$. Take the Job-GBERT model as an example, because it gets the best scores for all metrics and for the German data translated by DeepL. The difference between the Job-GBERT model on the English and the German data translated by DeepL is quite large, almost getting a 7% score distance. I assume the main reason for this phenomenon is that Job-GBERT (and also GBERT) model is pre-trained in German.

While we look at the German versions data, it is clear that the Job-GBERT model trained with DeepL translated data gets the highest scores and lowest Hamming loss on all the evaluation metrics. This result is understandable. In the previous section 3.2.3 'Machine Translation', I already explained that German data translated by DeepL API has a higher quality as it is a more mature translation engine compared with SJMM. So, all the models' performance on the DeepL translated German data is better than SJMM translated one. Furthermore, in another comparison, both job domain-adapted models outperform the general domain models in all the settings: Job-GBERT model outperforms the GBERT model, and Multi-Job-RoBERTa outperforms the Multi-BERT model. This is a very valuable and interesting phenomenon. Although the field of O*NET is similar to the field of job advertisements, there are some differences: 1) O*NET data is based on the USA job market, not Switzerland; 2) The text types are different. O*NET is an ontology compiled by experts, rather than job advertisements written by individual companies. Considering the good performance of the Job-GBERT model based on these gaps, I am hopeful and excited about the application of the Job-GBERT model to job advertisement data.

Another interesting point is that the both multilingual models seems to handle the SJMM-translated data better than other models. I assume this is because SJMM engine's specificity: it keeps the original English text when it's hard to translate. So the result is the final output contains two languages.

As discussed in 4.3.1, with the micro strategy, the accuracy, f1-score, recall, and precision have exactly identical numbers. Therefore, in Table 5.1 I only report the Micro-F1 score for the micro strategy. However, for the next step, I applied the

Job-GBERT model on the other two hierarchical levels with DeepL translated data. Table 5.2 reports the Job-GBERT model's results on the GWA, IWA and DWA levels with different weighted strategies for computation.

Level	Weighted Strategy	Accuracy	F1-score	Recall	Precision	Hamming_loss
GWA	macro	0.5819	0.5068	0.5103	0.5208	0.4181
	micro	0.5819	0.5819	0.5819	0.5819	0.4181
	support	0.5819	0.5709	0.5819	0.5670	0.4181
IWA	macro	0.413	0.3048	0.3107	0.3313	0.587
	micro	0.413	0.413	0.413	0.413	0.587
	support	0.413	0.4004	0.413	0.4147	0.587
DWA	macro	0.2151	0.0988	0.1353	0.0953	0.7849
	micro	0.2151	0.2151	0.2151	0.2151	0.7849
	support	0.2151	0.1711	0.2151	0.1694	0.7849

Table 5.2: Job-gbert model's performance on the IWA and DWA levels with DeepL translated training data. *support* refers to the 'support-weighted' strategy.

Comparing the results at three levels, it is noticeable that on the GWA level, the model gets the highest scores comprehensively and lower Hamming loss, and IWA level next, the GWA level worst. This result is consistent with our intuition, as the number of labels at the GWA level (37) is much lower than at the IWA level (332) and DWA level (2085), and therefore less difficult to classify.

5.1.1 Example Analysis

In order to understand the baseline models' performance better, especially about the multi-label situation, I select a task (ID: 8783) as an example for further analysis.

Table 5.3 gives the details about a multi-labeled task in the test set, including its three different GWA classes (GWA Title), and the translated GWA label by SJMM system (GWA_de) and DeepL API (GWA_deepl_de).

In this table, we can clearly see the difference differences between the two systems translating ontology data from English to German. Compared to the DeepL API, the SJMM system has a obvious under-translation issue who use the original English text directly when meet the text is hard to translate (see red text in table 5.3). The problem with this solution is that the final translation contains two languages: German and a small part of English. This situation is kind of proved by models' performance on test dataset.

In Table 5.1, when trained with German ontology data translated by the DeepL API, the Multi-BERT model performed worse than the other three models, while

when trained with data translated by the SJMM system, the performance of these models has no significant difference.

Task	GWA Title	GWA_sjmm_de	GWA_deepl_de
(8783) Wrap, weigh, label, and price cuts of meat. (DE-sjmm) Wrap, Gewichte, Label- und Preissenkungen von Meatas. (DE-deepl) Fleischstücke verpacken, wiegen, etikettieren und auszeichnen.	(4.A.3.a.2) Handling and Moving Objects.	Handling and Moving Objects.	Handhabung und Bewegung von Objekten
	(4.A.1.b.1) Identifying Objects, Actions, and Events.	Identifizierung von Objects, Actions- und Events.	Identifizierung von Objekten, Aktionen und Ereignissen.
	(4.A.1.b.3) Estimating the Quantifiable Characteristics of Products, Events, or Information.	Schätzen der quantitativen Eigenschaften von Produkten, Events, oder Informationen.	Schätzung der quantifizierbaren Merkmale von Produkten, Ereignissen oder Informationen.

Table 5.3: A task with multi-labels in test set

Table 5.4 and Table 5.5 give the GWA title predicted by four models trained with two versions of German data respectively. In brackets are the probability values given by the models for that prediction. All the predictions are sorted by the value of probabilities, and these predictions with the highest probability are considered as the models' final decisions for calculating the automated metrics. As task 8783 has more than one correct class (multi-label task), in the following tables, I highlighted all the correct predictions as blue in the top 5 answers provided by each model.

5.2 Hierarchical Classification Results

5.2.1 O*NET Data

As mentioned about before, all the hierarchical models only experiment on the DeepL-translated German version of O*NET data. In this part, when we talk about O*NET data, it refers only to this particular version.

Table 5.6 shows the RNN, RCNN and HMCN models' performance on the whole O*NET data set within a maximum of 50 training epochs (except the last row that shows the HMCN trained on 100 epochs). According to our principles, while we compare several models' performance, only the micro-f1 scores matter most and other scores are for reference only. Therefore, on the complete O*NET ontology set, the TextRNN model with recursive regularization reached the highest micro-f1 score: 0.3038 in 50 training epochs. This configuration is also used as the best hierarchical model to predict Job Ads data. In the following discussion, I used *'hierar'* to refer to the configuration that models are trained with recursive regularization, while *'flat'* means not.

Model	pred_la1	pred_la2	pred_la3	pred_la4	pred_la5
GBERT	Schätzen der quantitativen Eigenschaften von Produkten, Events, oder Informationen (0.516)	Dokumentation/ Recording Information. (0.0755)	Beschlüsse fassen und Probleme lösen. (0.0641)	Processing information. (0.0634)	Handling and Moving Objects. (0.0586)
Job-GBERT	Monitoring and Controlling Resources. (0.2376)	Schätzen der quantitativen Eigenschaften von Produkten, Events, oder Informationen (0.2118)	Processing information. (0.1299)	Handling and Moving Objects. (0.0866)	Beschlüsse fassen und Probleme lösen. (0.0831)
Multi-BERT	Monitoring and Controlling Resources. (0.567)	Handling and Moving Objects. (0.17)	Beschlüsse fassen und Probleme lösen. (0.1446)	Schätzen der quantitativen Eigenschaften von Produkten, Events, oder Informationen. (0.0288)	Processing information. (0.0182)
Job-Multi-BERT	Schätzen der quantitativen Eigenschaften von Produkten, Events, oder Informationen. (0.4998)	Handling and Moving Objects. (0.2074)	Identifizierung von Objects, Actions- und Events. (0.075)	Beschlüsse fassen und Probleme lösen. (0.0547)	Processing information. (0.0324)

Table 5.4: An examples of baseline prediction on GWA level. Models are trained with data translated by SJMM system.

For a better understanding of how models are trained, Figure 5.1 visualized the micro-f1 score during the training process of TextRNN_flat, TextRNN_hierar, TextRCNN_flat, TextRCNN_hierar.

Combining Table 5.6 and Figure 5.1, we can find that *'flat'* models have a tendency to perform best out of shorter epochs compared to the *'hierar'* models. This situation can be explained by the recursive regularization technique which is only applied to the *'hierar'* models. This regularisation technique limits the extent to which the model parameters are updated during training and in this way controls overfitting. Take a look at the graph 5.1, which also supports the hypothesis that the *'hierar'* model achieves around 0.8 on the training set, while the *'flat'* model achieves a figure of around 0.9.

However, what is more important is that compared to *'flat'* models, *'hierar'* models can achieve higher f1-scores in the same 50 training epochs. For the TextRCNN

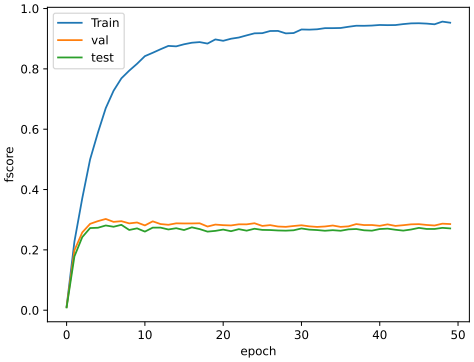
Model	pred_la1	pred_la2	pred_la3	pred_la4	pred_la5
GBERT	Handhabung und Bewegung von Objekten. (0.7288)	Schätzung der quantifizierbaren Merkmale von Produkten, Ereignissen oder Informationen. (0.0934)	Identifizierung von Objekten, Aktionen und Ereignissen. (0.0442)	Beurteilung der Qualitäten von Objekten, Dienstleistungen oder Personen. (0.014)	Ausführen allgemeiner körperlicher Aktivitäten. (0.0136)
Job-GBERT	Dokumentieren /Aufzeichnen von Informationen. (0.2876)	Identifizierung von Objekten, Aktionen und Ereignissen (0.2765)	Handhabung und Bewegung von Objekten. (0.2252)	Schätzung der quantifizierbaren Merkmale von Produkten, Ereignissen oder Informationen. (0.0484)	Kreativ denken. (0.0302)
Multi-BERT	Identifizierung von Objekten, Aktionen und Ereignissen. (0.5779)	Handhabung und Bewegung von Objekten. (0.1827)	Verarbeitung von Informationen. (0.0435)	Durchführung von Verwaltungstätigkeiten. (0.0359)	Dokumentieren /Aufzeichnen von Informationen. (0.0258)
Job-Multi-BERT	Handhabung und Bewegung von Objekten. (0.6087)	Identifizierung von Objekten, Aktionen und Ereignissen. (0.1878)	Schätzung der quantifizierbaren Merkmale von Produkten, Ereignissen oder Informationen. (0.038)	Durchführung von Verwaltungstätigkeiten. (0.0268)	Dokumentieren /Aufzeichnen von Informationen. (0.0233)

Table 5.5: An examples of baseline prediction on GWA level. Models are trained with DeepL translated data.

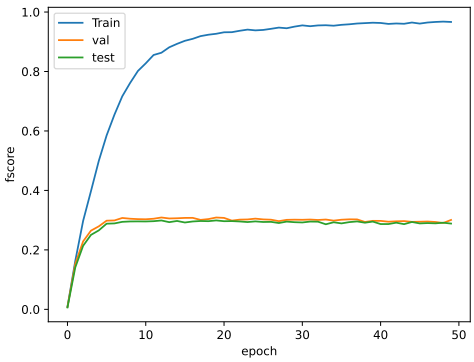
Model	Conf	Precision	Recall	Micro-f1	Macro-f1	Time	Best
TextRNN	flat	0.3952	0.2179	0.2809	0.0499	55	6
	hierar	**0.4740**	0.2235	**0.3038**	0.0591	100	31
TextRCNN	flat	0.3494	0.2612	0.2989	0.0801	55	13
	hierar	0.2463	**0.3781**	0.2983	**0.0999**	110	27
HMCN	-	0.4142	0.2307	0.2963	0.0611	65	43
	-	0.3724*	0.2569*	**0.3041***	0.0877*	65	86(-100)

Table 5.6: Models' results on the whole O*NET ontology dataset. *Precision* and *Recall* are calculated with micro-strategy. *Time* refers to the seconds cost each epoch. *Best* is the epoch that gets the best performance during 50 training epochs. * means this score is got within a maximum of 100 training epochs, instead of 50.

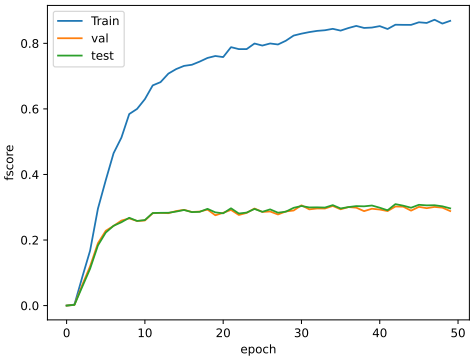
model, 'RCNN_flat' and 'RCNN_hierar' achieve nearly identical scores. The 'TextRNN_hierar' gets 2 percentage points higher f1-score compared with 'RNN_flat', which is also the highest score among all the models.



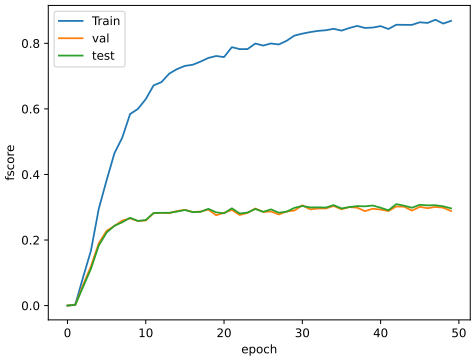
(a) RNN_flat



(b) RCNN_flat

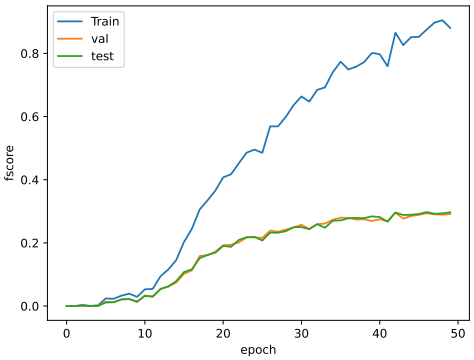


(c) RNN_hierar

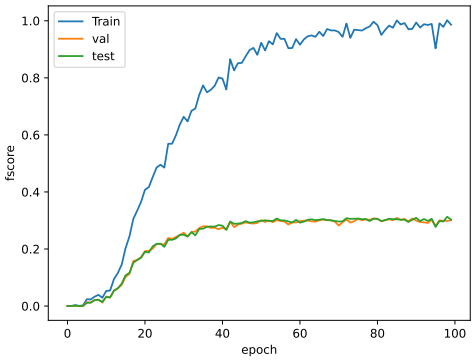


(d) RCNN_hierar

Figure 5.1: Micro-F1 score visualization during training process.



(a) Micro-F1 for 50 epochs



(b) Micro-F1 for 100 epochs

Figure 5.2: Micro-F1 score visualization of HMCN model.

Figure 5.2a shows the micro-f1 scores of the HMCN model during 50 epochs. It can be seen that the f1-score pattern of the HMCN model during training differs from that of TextRNN and TextRCNN models. Overall, its f1-score climbs more slowly.

It was not until the 43/50th epoch that the best score was achieved. Therefore, I continue to train the HMCN model with identical parameters over 100 epochs. As I expected, the best F1-score over 100 epochs improved by nearly 1 percent point compared to 50 training epochs. 0.3041 is also the highest f1-score I get on the complete O*NET ontology set. Meanwhile, as shown in Figure 5.2b, the F1-score of HMCN has stabilized after 100 training epochs.

5.2.2 On the Subset

Table 5.7 gives the results of HiAGM-LA with Tree-LSTM as encoder on three subsets. In this table, the HiAGM-LA model gets the highest precision, recall, and micro-f1 scores on subset3 and the best macro-f1 score on subset2. There are several possible reasons for this situation: i) Sub1 is more structurally complex because it has nearly 900 child labels for 5 GWA labels. Therefore, it is more difficult to classify compared with sub3. ii) There are only 300 training samples in sub2 resulting in the model not being fully trained on this subset.

Subset	Precision	Recall	Micro-F1	Macro-F1
sub1	0.4240	0.2927	0.3463	0.0737
sub2	0.4710	0.2655	0.3395	**0.1505**
sub3	**0.4862**	**0.3694**	**0.4198**	0.0838

Table 5.7: HiAGM-LA results on three O*NET subsets. *Precision* and *Recall* are calculated with micro-strategy.

In Table 5.8, an overview of the model's performance on O*NET subset 3 is reported, with several interesting facts. First, the HiAGM model was successfully experimented with on the subset in four different configurations. HiAGM model integrates the encoded hierarchical information with textual input and therefore takes much longer compared to the other three models. Although the difference in time cost for different configurations is almost consistent with that reported by the authors in their previous study, the results of HiAGM on the sub-O*NET training set are somewhat different from what they reported on the RCV1_V2. The highest micro-f1 score was obtained for HiAGM-TP with the Tree-LSTM encoder, but not for HiAGM-TP with the GCN encoder. Furthermore, although HiAGM reports better results on public datasets (e.g. RCV1), it can not outperform others on the subset of O*NET ontology data.

Second, experiments on the subset further demonstrate the effectiveness of the recursive regularization technique. For both the TextRNN and TextRCNN model,

configuring with 'hierar' get higher scores than 'flat' comprehensively. And the TextRCNN_hierar gets the highest micro-f1 score 0.5526 among all the models and configurations.

Third, the HiAGM, HMCN, TextRNN and TextRCNN models employ different strategies for precision and recall trade-offs. Some of these models (e.g. TextRCNN) have more of a trade-off between recall and precision, while others (e.g. TextRNN) seem to focus more on precision. TextRNN_hierar receives the highest precision score of 0.722, which is 0.24 points higher than the lowest HiAGM-TP model with GCN encoder. However, recall is also an important metric for evaluating model performance in this project, so the TextRCNN model, which focuses more on the balance of precision and recall, receives the best micro-F1 scrore 0.5526, and TextRCNN_hierar is considered to be the best configuration.

Last, notably, all the models that experiment on both complete and O*NET subset, achieve higher scores on the subset. I assume that this situation is because the larger hierarchical label structure contains more semantically similar labels, which makes accurate classification more difficult.

Model	Conf	Precision	Recall	Micro-F1	Macro-F1	Time	Best
TextRNN	flat	0.6583	0.3893	0.4893	0.0949	5	88
	hiear	**0.7220**	0.3976	0.5128	0.0835	7	42
TextRCNN	flat	0.6409	0.4597	0.5354	0.1082	5	20
	hiear	0.6794	**0.4656**	**0.5526**	**0.1156**	8	19
HiAGM-LA	tree	0.4862	0.3694	0.4198	0.0838	160	78
	gcn	0.5404	0.3639	0.4350	0.0823	60	53
HiAGM-TP	tree	0.5493	0.3701	0.4422	0.0840	170	96
	gcn	0.4860	0.3853	0.4299	0.0876	170	95
HMCN	-	0.6483	0.4308	0.5176	0.0943	5	86

Table 5.8: Models' results on the O*NET subset. *precision* and *recall* are calculated with micro-strategy. *Time* refers to the seconds cost each epoch. *Best* is the epoch get best performance during 100 training epochs.

5.3 Reference on the Job Ads Data

The Job Ads data is predicted by selected baseline and global hierarchical models separately.

5.3.1 Baseline Model's Predictions

From the four baseline models, the Job-GBERT model gets the highest scores on DeepL training data, so it is selected to infer the job ads data on three hierarchical levels separately. A German-speaking domain expert manually evaluated the top 3 predictions of 200 samples on 3 hierarchical levels (4.1.1.2 gives the information about the Job Ads data). Overall, 1800 ($3 \times 3 \times 200$) model predictions are evaluated and another 197 samples with the top 3 predictions made by the best hierarchical model are evaluated. Table 4.3 shows an example that contains the prediction and evaluation details. The overall evaluation results are in Table 5.9. *Precision* and *Precision_soft* mean the percentages of samples in the first prediction (highest probability) that are 'totally right' and 'acceptable' respectively. Similarly, *Precision_soft* and *Precision_t3_soft* refer to the percentages in the top three predictions. *scores_t3* is the average precision score for the top three predictions and considers the 'acceptable' predictions as wrong, while *scores_t3_soft* is a more lenient metric that considers all 'acceptable' predictions as right. And both two precision scores employ 'samples' as the average strategy.

Level	Precision	Precision_soft	Precision_t3	Precision_t3_soft	scores_t3	scores_t3_soft
GWA	**0.5909**	**0.7929**	0.7424	**0.9394**	**0.9811**	**0.9705**
IWA	0.5859	0.7879	**0.7525**	0.9242	0.9662	0.9598
DWA	0.4697	0.6717	0.5909	0.8081	0.9725	0.9577

Table 5.9: Job ads evaluation statistics. *-soft* means considering the 'acceptable' predictions as 'right'. *-t3* means considering the top 3 predictions.

Surprisingly, we find that the trained Job-GBERT model performs much better on the Job Ads dataset than the O*NET ontology set. Even though looking at the most strict metric *'Precision'*, it reaches 0.5909, 0.5859 and 0.4697 on GWA, IWA and DWA levels respectively, which are all higher than that at the O*NET data set (micro-f1 scores: 0.5819, 0.413 and 0.2151, see Table 5.2). To be more specific, this gap between two data sets increases the lower down the classification level it goes: 0.08 at the GWA level, 0.1749 at the IWA level and 0.2547 at the DWA level. I assume that this is because our domain expert does not exactly classify on the same criteria as the O*NET. In O*NET dataset, most Tasks are assigned to only one (14752/15721, see Table 3.5) or two DWA labels (of course, also their corresponding parent IWA and GWA labels), but some other Work Activities (WA) seem also be 'right' or 'acceptable'.

Furthermore, the gap between GWA and IWA is much smaller than that between IWA and DWA. Except for the *scores*, all the gaps between GWA and IWA levels are less than 1 percent point, while the gaps normally achieve 10 percent points for

IWA and DWA levels. I assume this is because the number of labels at the GWA level (37) and IWA level (332) are less than that at the DWA level (2085). Due to the huge number of labels, there are more semantically similar labels at the DWA level which is quite difficult to classify.

5.3.2 Global Hierarchical Model's Predictions

RNN_hierar model outperforms all the global hierarchical models (see Table 5.6) in 50 training epochs, so it is selected to predict the Job Ads data. The biggest difference compared to the previous baseline model is that global hierarchical models predict all-layer labels together. We manually evaluate the top 3 predictions for 197 samples in job ads data, so there are 591 (3×197) evaluated predictions. Table 5.10 reports its results based on our annotator's evaluation.

There are several interesting points of global hierarchical models' prediction. First, the overall scores are lower than the baseline models' predictions even compared with the DWA level. This result is contrary to our initial expectations: with hierarchical information, global hierarchical models are expected to outperform baseline models. Several points may contribute to this situation:

1. Global hierarchical models put all levels of labels together for prediction, rather than concentrating on one particular level. Therefore, one global model needs to consider more labels at the same time, which leads to even worse results.

2. The RNN_hierar model is a general domain model, whereas Job-GBERT is pre-trained with job domain data. This is why the Job-GBERT model gives better results on Job Ads data than on the O*NET dataset. This comparison also shows the effectiveness of the domain-adapted technique.

Second, the majority of predictions from the RNN_hierar model are GWA-level labels (468/591). This is because all predictions are ranked according to the likelihood scores calculated by the model. Because there are only 37 GWA labels with the same amount of training data, but over 2000 child-labels at the IWA and DWA levels, it is reasonable for the model to have more confidence in the GWA labels. Therefore, the GWA labels appear more frequently in the first 3 predictions made by the RNN_hierar model.

Model	Precision	Precision_soft	Precision_t3	Precision_t3_soft
RNN_hierar	0.3198	0.467	0.5228	0.7259

Table 5.10: Statistics of RNN_hierar predictions of Job Ads data. *-soft* means considering the 'acceptable' predictions as 'right'. *-t3* means considering the top 3 predictions.

Task	Model prediction	Eval score
Verpackungsarbeiten allgemein und in Medizinaltechnik	Anderen helfen und für sie sorgen	0
	Analysieren von Daten oder Informationen	0
	Kommunikation mit Personen au erhalb der Organisation	0
Planung Durchführung und Auswertung von hochauflösenden 2D und 3D seismischen Messungen zur Charakterisierung potentieller Standorte für geologische Tiefenlager	Analysieren von Daten oder Informationen	1
	Kreativ denken	0.5
	Analysieren von Daten oder Informationen–Analysieren Sie Daten um den Betrieb zu verbessern	0.5
sortieren	Verarbeitung von Informationen–Materialien oder Produkte sortieren	1
	Verarbeitung von Informationen–Materialien oder Produkte sortieren–Sortieren von Materialien oder Produkten zur Verarbeitung Lagerung zum Versand oder zur Sortierung	1
	Reparatur und Wartung von mechanischen Geräten	0
Daten	Handhabung und Bewegung von Objekten	×
	Informationen erhalten	0
	Dokumentieren Aufzeichnen von Informationen	×

Table 5.11: Examples of Job Ads with the RNN_hierarchy model's predictions

5.4 O*NET Ontology Data Difficulties Analysis

For these two series of experiments we did on the O*NET dataset, the result scores (see Table 5.1 and Table 5.6) are relatively lower than scores reported on public datasets (like RCV1) by previous studies 2.2). For example, the HMCN model achieves 83.96 on the RCV1-V2 dataset (Mao et al. [2019]), however, the score on the complete O*NET ontology set is only 30.41. Therefore, I did some analysis to find out the stumbling blocks for global hierarchical models to classify successfully on the O*NET dataset.

5.4.1 Complex Ontology Hierarchy

First of all, I pay attention to the hierarchy structure which is the main heart of this task. In almost all the previous studies, the key to their methods is successfully making use of hierarchical information, in different ways though.

However, O*NET ontology hierarchical structure is much more complex than most public datasets. Table 5.12 compares the statistical characters of the O*NET dataset with other public datasets that are mentioned in the section 2.1. It is clear that the number of classes in the O*NET ontology set is much more than other sets and the number of training samples also less than others, which makes it considerably more difficult to classify.

Data Set	$\|C\|$	Depth	Avg($\|C_i\|$)	Train	Val	Test
RCV1-V2	103	4	3.24	20,833	2,316	781,265
WOS	141	2	2	30,070	7,518	9,397
NYT	166	8	7.6	23,345	5,834	7,292
O*NET	2454	3	3.97	16,480	2,354	4,708

Table 5.12: O*NET dataset statistics comparison. $|C|$ is the number of classes. *Depth* is the maximum level of hierarchy. *Avg($|C_i|$)* is the average number of classes per sample. *Train/ Val/ Test* are the size of the train/validation/test set respectively.

5.4.2 High Semantic Similarity of Text & Classes

There exist many semantically similar classes. And this phenomenon is more pronounced at the level further down the hierarchy.

In Table 5.13, there are five tasks and the DWA labels that they are assigned to in the O*NET test set. All these DWA labels belong to one IWA label. These labels are quite difficult to distinguish even for humans. For example, Task1 is only assigned to DWA *'Direct organizational operations, activities, or procedures'*, but DWA3 *'Direct organizational operations, projects, or services'* also seems right. Also, Task2 *'Inspect meals served for conformance to prescribed diets and standards of palatability and appearance'* which is only assigned to DWA2 *'Manage preparation of special meals or diets'* is also suitable for DWA5 *'Manage food service operations or parts of operations'*. Accurate classification in this case requires too much of the model to extract the semantics of the text, so the overall performance of both types of models (baseline and global hierarchical models)on the O*NET dataset is not as high as other public datasets.

While we are manually evaluating job tasks from job ads, this kind of situation may be classified as *1, 'totally right'* or *0.5, 'acceptable'*. This is also one of the reasons why the Job-GBERT model gets much higher on the Job Ads data than the O*NET dataset.

Number	Task	DWA Title	IWA Title
1	Determine response requirements and relative priorities of situations, and dispatch units in accordance with established procedures.	Coordinate operational activities.	Direct organizational operations, activities, or procedures.
2	Inspect meals served for conformance to prescribed diets and standards of palatability and appearance.	Manage preparation of special meals or diets.	
3	Coordinate activities of departments, such as sales, graphic arts, media, finance, and research.	Direct organizational operations, projects, or services.	
4	Route proofs with marked corrections to authors, editors, typists, or typesetters for correction or reprinting.	Coordinate operational activities.	
5	Plan, organize, and control the operations of a cocktail lounge or bar.	Manage food service operations or parts of operations.	

Table 5.13: Examples of tasks that are assigned to semantically similar DWA labels.

6 Conclusion

In summary, it is feasible to apply a hierarchy-aware classification and cross-linguistic approaches to classify job tasks from job advertisements into O*NET ontology data.

In this project, I designed two series of experiments to achieve this goal. First, transformer-based classification models were employed independently at each level, working as baseline models. In this part, four different models were trained on three different versions of O*NET data. And these experiments demonstrated the following points:

1. Machine translation methods are more effective than multilingual models when dealing with language gaps between ontology data and job advertisement data.
2. Domain adaptation is an effective mechanism. Two of the four baseline models that are pre-trained with job domain data both perform better than the corresponding general domain models.
3. The quality of the translated text affects the performance of the model classification. All transformer-based models perform better on the German O*NET data translated by DeepL compared to the data translated by the SJMM engine.

Second, some state-of-the-art models, such as the TextRNN, TextRCNN, HMCN and HiAGM, are used as global hierarchical models to classify the tasks. This series of experiments contribute to this goal in several ways:

1. Using hierarchical information. These global models mentioned above deal with hierarchical information in different ways, however, they all treat the hierarchy as a whole, rather than building separate classifiers for each level or node.
2. Demonstrating the effectiveness of recursive regularization which is a widely used technique for hierarchical classification task. The use of hierarchical penalty during training avoids common inconsistencies and overfitting problems.

Last but not least, the best configurations from these two series of experiments on O*NET data are selected to predict the Job Ads data separately and achieve the final classification. Relying on the annotation of these predictions by our German-speaking domain expert, we demonstrate that our models are able to provide reliable classification of Job Ads data in the O*NET hierarchical ontology.

The specificity of this task makes it much harder than other similar hierarchical classification problems: i) the gap between the training data and the final inference data. Since there is no existing Job Ads data available for model training, we can only train the model on a similar O*NET dataset. However, different regions (Switzerland and USA), and different languages (German and English) all add to the difficulty of the task. ii) Complex hierarchy of O*NET data. *task* data in the O*NET ontology set is not only hierarchical but also multi-label in the sense that one *task* can belong to several work activities. Combined with this the large number of the classes, and the task becomes even more difficult. iii) The similarity between *tasks* and classes in the O*NET data makes it difficult for the model to correctly classify exactly according to the O*NET standard.

By comparing the performance of the two families of models on the job advertisement dataset, it can be seen that applying global hierarchical models does not improve the quality of the task classification, although these models report very good scores on some public datasets. In the following SJMM study, I recommend using the best baseline model Job-GBERT for the job advertisements classification task. Compared to the state-of-the-art hierarchical classification models, the transformer-based Job-GBERT model's specificity is more relevant to our task, because this model is pre-trained using data from within the German language and advertising domains. However, in this way, extra attention should be paid to inconsistency that could result from building local classifiers at each level.

6.1 Future Work

For future work, there are three points that can be considered in further research to improve classification performance.

First, future work could make use of the semantics of the labels. In this study, I processed all WA labels numerically, ignoring the semantics in the tags. However, considering that some labels, especially low-level tags like DWA, are even longer than the extracted work task, it is interesting to examine these labels.

Second, in the current study, only one German-speaking domain expert annotated

the first three predictions of a sample of approximately 200 job advertisements. Naturally, it would be better to have more annotators to annotate the job advertisement data. In this way, the inter-annotator agreement (IAA) can also be calculated to assess the quality of the predictions.

Finally, the Job Ads data used in this project is based on the previous zoning task of Gnehm and Clematide [2020], so *task* is a bit over-processed for this hierarchical classification problem. In some cases, *task* is even shorter than its label's text (see Table 5.11).

References

M. Artetxe, S. Ruder, and D. Yogatama. On the Cross-lingual Transferability of Monolingual Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://aclanthology.org/2020.acl-main.421.

J. Blitzer, R. McDonald, and F. Pereira. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006. URL https://aclanthology.org/W06-1615.

B. Chan, S. Schweter, and T. Möller. German's Next Language Model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.598. URL https://aclanthology.org/2020.coling-main.598.

H. Chen, Q. Ma, Z. Lin, and J. Yan. Hierarchy-aware Label Semantics Matching Network for Hierarchical Text Classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4370–4379, 2021.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. URL https://aclanthology.org/2020.acl-main.747.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski. Hierarchical annotation of medical images. *Pattern Recognition*, 44(10-11):2436–2449, 2011.

K. Duh, A. Fujino, and M. Nagata. Is Machine Translation Ripe for Cross-lingual Sentiment Classification? In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 429–433, 2011.

A.-S. Gnehm. Text Zoning for Job Advertisements with Bidirectional LSTMs. In *Proceedings of the 3rd Swiss Text Analytics Conference- Swiss-Text 2018*, pages 66–74. CEUR Workshop Proceedings, 2018.

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.

A.-s. Gnehm, E. Bühlmann, H. Buchs, and S. Clematide. Fine-Grained Extraction and Classification of Skill Requirements in German-Speaking Job Ads. In *Proceedings of the Fifth Workshop on Natural Language Processing and Computational Social Science (NLP+CSS)*, pages 14–24, Abu Dhabi, UAE, Nov. 2022a. Association for Computational Linguistics. URL https://aclanthology.org/2022.nlpcss-1.2.

A.-S. Gnehm, E. Bühlmann, and S. Clematide. Evaluation of Transfer Learning and Domain Adaptation for Analyzing German-speaking Job Advertisements. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3892–3901, 2022b.

S. Gopal and Y. Yang. Recursive Regularization for Large-scale Classification with Hierarchical and Graphical Dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 257–265, 2013.

R. Johnson and T. Zhang. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1011.

K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, , M. S. Gerber, and L. E. Barnes. HDLTex: Hierarchical Deep Learning for Text Classification. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. IEEE, 2017.

S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

D. D. Lewis, Y. Yang, T. Russell-Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

L. Liu, F. Mu, P. Li, X. Mu, J. Tang, X. Ai, R. Fu, L. Wang, and X. Zhou. NeuralClassifier: an Open-source Neural Hierarchical Multi-label Text Classification Toolkit. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–92, 2019.

P. Liu, X. Qiu, and X. Huang. Recurrent Neural Network for Text Classification with Multi-Task Learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, page 2873–2879. AAAI Press, 2016.

S. D. Magaldi. A Task-specific Neural Translation Model for the Stellenmonitor Pipeline (Bachelor's Thesis). University of Zurich: Institute of Computational Linguistics., 2021.

Y. Mao, J. Tian, J. Han, and X. Ren. Hierarchical Text Classification with Reinforced Label Assignment. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 445–455, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1042. URL `https://aclanthology.org/D19-1042`.

R. Mihalcea, C. Banea, and J. Wiebe. Learning Multilingual Subjective Language via Cross-lingual Projections. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 976–983, 2007. URL `https://aclanthology.org/P07-1123`.

F. E. Otero, A. A. Freitas, and C. G. Johnson. A Hierarchical Multi-label Classification ant Colony Algorithm for Protein Function Prediction. *Memetic Computing*, 2:165–181, 2010.

T. Pires, E. Schlinger, and D. Garrette. How Multilingual is Multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1493. URL https://aclanthology.org/P19-1493.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based Learning of Hierarchical Multilabel Classification Models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

K. S. Tai, R. Socher, and C. D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1150. URL https://aclanthology.org/P15-1150.

C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision Trees for Hierarchical Multi-label Classification. *Machine learning*, 73:185–214, 2008. URL https://doi.org/10.1007/s10994-008-5077-3.

X. Wan. Co-training for Cross-lingual Sentiment Classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243, 2009.

Z. Wang, P. Wang, L. Huang, X. Sun, and H. Wang. Incorporating Hierarchy into Text Encoder: a Contrastive Learning Approach for Hierarchical Text Classification. pages 7109–7119, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.491. URL https://aclanthology.org/2022.acl-long.491.

J. Wehrmann, R. Cerri, and R. Barros. Hierarchical Multi-Label Classification Networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5075–5084. PMLR, 10–15 Jul 2018.

J. Wu, W. Xiong, and W. Y. Wang. Learning to Learn and Predict: A Meta-Learning Approach for Multi-Label Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4354–4364, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1444.

S. Wu and M. Dredze. Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China, 2019.

J. Zhou, C. Ma, D. Long, G. Xu, N. Ding, H. Zhang, P. Xie, and G. Liu. Hierarchy-aware Global Model for Hierarchical Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1106–1117, 2020. URL `https://aclanthology.org/2020.acl-main.104`.

A Appendix

Parent Node	Children Nodes
Root	CCAT ECAT
CCAT	C12 C13 C15 C18 C22 C24 C31 C33 C41
C15	C151 C152
C151	C1511
C18	C181
C31	C311
C41	C411
ECAT	E12 E21 E51 E71
E21	E211 E212
E51	E512

Table 1: Sample taxonomy structure of RCV1 dataset.

Level	Weighted Strategy	Accuracy	F1-score	Recall	Precision	Hamming_loss
GBERT	macro	0.5458	0.4751	0.4803	0.4888	0.4542
	micro	0.5458	0.5458	0.5458	0.5458	0.4542
	weighted	0.5458	0.5252	0.5458	0.5158	0.4542
Job-GBERT	macro	0.533	0.4535	0.4507	0.4879	0.467
	micro	0.533	0.533	0.533	0.533	0.467
	weighted	0.533	0.5134	0.533	0.5078	0.467
Multi-BERT	macro	0.5317	0.4371	0.4347	0.4845	0.4683
	micro	0.5317	0.5317	0.5317	0.5317	0.4683
	weighted	0.5317	0.5086	0.5317	0.5068	0.4683
Multi-Job-RoBERTa	macro	0.5453	0.4549	0.4524	0.488	0.4547
	micro	0.5453	0.5453	0.5453	0.5453	0.4547
	weighted	0.5453	0.5311	0.5453	0.5308	0.4547

Table 2: All metrics of models trained on the SJMM translated O*NET data.

Table 2 shows four baseline models' results on the German O*NET data which is translated by the SJMM system. For more information, all the metrics are computed with three different average strategies.

Level	Weighted Strategy	Accuracy	F1-score	Recall	Precision	Hamming_loss
GBERT	macro	0.5804	0.5037	0.5089	0.5074	0.4196
	micro	0.5804	0.5804	0.5804	0.5804	0.4196
	weighted	0.5804	0.5690	0.5804	0.5645	0.4196
Job-GBERT	macro	0.5819	0.5068	0.5103	0.5208	0.4181
	micro	0.5819	0.5819	0.5819	0.5819	0.4181
	weighted	0.5819	0.5709	0.5819	0.5670	0.4181
Multi-BERT	macro	0.5487	0.4704	0.4801	0.481	0.4513
	micro	0.5487	0.5487	0.5487	0.5487	0.4513
	weighted	0.5487	0.5379	0.5487	0.5391	0.4513
Multi-Job-RoBERTa	macro	0.5763	0.4862	0.4866	0.5057	0.4237
	micro	0.5763	0.5763	0.5763	0.5763	0.4237
	weighted	0.5763	0.5646	0.5763	0.5631	0.4237

Table 3: All metrics of models trained on the DeepL translated O*NET data.

Table 3 gives all the metrics of four baseline models on the DeepL translated German O*NET data.