Extraction of Skills and Benefits from Job Postings Descriptions

Andrei Shchapaniak
ČVUT - FIT
shchaand@fit.cvut.cz

December 13, 2024

1 Introduction

This research focuses on the automated extraction of skills and benefits from job postings, with an emphasis on minimizing overlapping identifications. The project encompasses both the development of an efficient extraction system and its optimization, validated through manual annotation of a selected job subset.

2 Input data

Three datasets were used to for this assignment. *Indeed Job Posting Dataset*[1] contains original job descriptions; *Job Dataset*[2] contains features with benefits from different job descriptions; and *ESCO Skills Dataset*[3] contains huge set of known skills. To enhance benefit detection capabilities, the initial set of 23 unique benefits was augmented with over 40 additional job benefits generated through ChatGPT, ensuring broader coverage of modern workplace offerings.

3 Methods

The study implements a dynamic n-gram approach for extracting skills and benefits from job descriptions. The system generates variable-length n-grams for each sentence and employs cosine similarity matching against predefined datasets, enabling flexible and context-aware identification of relevant information.

The research proceeded in two phases. The first phase evaluated four Large Language Models (**LLMs**) - JobBert, DistilBERT, MiniLM, and Roberta - selected for their proven capabilities in text processing and information extraction.

[1]https://www.kaggle.com/datasets/promptcloud/indeed-job-posting-dataset/data
[2]https://www.kaggle.com/datasets/ravindrasinghrana/job-description-dataset
[3]https://www.kaggle.com/datasets/thenoob69/esco-skills

`JobBERT`, specifically trained on job posting data, demonstrated superior performance in skill extraction tasks as shown in the SkillSpan study [4], making it particularly relevant for our job-focused analysis. `DistilBERT`, known for its efficient architecture that retains 95% of BERT's performance while reducing the model size by 40% [2], and `RoBERTa`, which has shown impressive results through its optimized training procedure [1], serve as strong benchmarks for comparison. `MiniLM` was included due to its demonstrated efficiency in task-agnostic compression while maintaining high performance [3].

The evaluation process involved manual annotation of 10 randomly selected job postings, with model performance assessed through F1 score, Accuracy, and Recall metrics.

3.1 Optimization strategy

The implementation incorporates three key optimization techniques:

- Embedding caching: Utilizes a Least Recently Used (`LRU`) cache (512-entry capacity) to store embeddings of frequently occurring text segments.
- Efficient n-gram generation: Employs a dictionary-based approach to eliminate redundant processing.
- Enhanced similarity calculations: Implements optimized vector operations including batched matrix multiplication for parallel similarity computation, memory-efficient tensor operations using `torch.no_grad()` and threshold-based filtering using tensor masks.

A useful approach in the embedding process is combining n-gram and sentence-level context through weighted embeddings, where each n-gram embedding is weighted at 80% and its parent sentence embedding at 20%. This weighting, determined through experimental testing, helps balance local phrase meaning with broader sentence context.

	Model	Accuracy	Recall	F1-score
Skills	MiniLM	**0.57**	**0.40**	**0.47**
	JobBert	0.12	0.30	0.17
	Roberta	0.08	0.20	0.11
	DistilBert	0.11	**0.40**	0.17
Benefits	MiniLM	**1.00**	**0.57**	**0.73**
	JobBert	0.67	**0.57**	0.62
	Roberta	0.44	**0.57**	0.50
	DistilBert	0.22	**0.57**	0.32

Table 1: Comparison of LLMs.

3.2 Data Preprocessing

Working with raw data is generally not considered good practice. Therefore, I preprocessed the datasets before running the pipeline with measurements to improve text processing performance. The preprocessing pipeline includes:

- Text normalization: newline removal, lowercase conversion.
- Stop word elimination using NLTK[4].
- HTML tag removal from job descriptions using BeautifulSoup[5].

4 Results

The experiments I conducted in Google Colab[6] using a Tesla T4 GPU runtime configuration. All subsequent results were obtained using a similarity threshold of 85% between n-grams and dataset items.

4.1 Comparison of LLMs

Based on the results from Table 1 above, `MiniLM` demonstrated the highest overall performance for both skills and benefits extraction, achieving the best F1-score in both categories (0.47 for skills and 0.73 for benefits). Although the dataset for benefits extraction is relatively small, which may have influenced the higher results and smaller differences, MiniLM still outperformed the other models significantly. Therefore, MiniLM will be selected for the next part of the study, focusing on optimizing its performance.

4.2 Extractor optimization

I conducted performance testing to evaluate the optimization effectiveness using a sample of 15 randomly selected job descriptions from the dataset. The optimized implementation completed the extraction process in 3.45 minutes, while the initial implementation required 4.23 minutes. This represents a **17%** reduction in processing time, demonstrating that our optimization strategies showed some improvement in the speed of skills/benfits extraction.

5 Conclusion

This analysis demonstrates that while unsupervised extraction based on cross-field similarity computations shows potential, it requires significant refinement for optimal performance in job description processing. A more effective approach would involve focusing on a limited domain to reduce terminology variance, incorporating Named Entity Recognition (`NER`) systems for precise entity identification, and leveraging larger volumes of labeled data to improve model accuracy. Nevertheless, MiniLM's performance with dynamic n-gram generation proved promising for unsupervised extraction scenarios where labeled data is scarce, suggesting its viability as a foundation for specialized information extraction systems when combined with domain-specific enhancements.

References

[1] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[2] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version

[4] https://pythonspot.com/nltk-stop-words/
[5] https://pypi.org/project/beautifulsoup4/
[6] https://colab.research.google.com

of bert: smaller, faster, cheaper and lighter, 2020.

[3] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[4] Mike Zhang, Kristian Nørgaard Jensen, Sif Dam Sonniks, and Barbara Plank. Skillspan: Hard and soft skill extraction from english job postings, 2022.