I.J. Intelligent Systems and Applications, **2025, 5, 95-157**
Published Online on October 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2025.05.07

Smart Application for Recruiting Based on Natural Language Processing Methods, Transformer Models and Siamese Neural Network Architecture

Diana Vyshotravka
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: vyshotravka.diana.sa.2021@lpnu.ua
ORCID iD: https://orcid.org/0009-0007-5616-2438

Victoria Vysotska
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: Victoria.A.Vysotska@lpnu.ua
ORCID iD: https://orcid.org/0000-0001-6417-3689

Zhengbing Hu
School of Computer Science, Hubei University of Technology, Wuhan, China
E-mail: drzbhu@gmail.com
ORCID iD: https://orcid.org/0000-0002-6140-3351

Dmytro Uhryn
Department of Computer Science, Educational and Research Institute of Physical, Technical and Computer Sciences, Yuriy Fedkovych Chernivtsi National University, 58012, Ukraine
E-mail: d.ugryn@chnu.edu.ua
ORCID iD: https://orcid.org/ 0000-0003-4858-4511

Yuriy Ushenko*
Department of Physics, Shaoxing University, Shaoxing, Zhejiang Province 312000, China
Department of Computer Science, Educational and Research Institute of Physical, Technical and Computer Sciences, Yuriy Fedkovych Chernivtsi National University, 58012, Ukraine
E-mail: y.ushenko@chnu.edu.ua
ORCID iD: https://orcid.org/0009-0002-7620-5355
*Corresponding author

Kyrylo Smelyakov
Department of Software Engineering, Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, 61166, Ukraine
E-mail: kyrylo.smelyakov@nure.ua
ORCID iD: https://orcid.org/0000-0001-9938-5489

Received: 06 June 2025; Revised: 28 July 2025; Accepted: 26 August 2025; Published: 08 October 2025

Abstract: This study presents a deep learning-based approach to automated resume and job matching that uses semantic similarity between texts. The solution is based on SimCSE RoBERTa transformer embeddings and a Siamese neural architecture trained using the MSELoss loss function. Unlike traditional filtering systems by keywords or characteristics, the proposed model learns to place semantically compatible pairs (resume-vacancy) in a common vector space. Unlike traditional keyword-based or attributive matching systems, our method is designed to capture deep semantic alignment between resumes and job descriptions. To evaluate the effectiveness of this architecture, we conducted extensive experiments on a labelled dataset of over 7,000 resume–vacancy pairs obtained from the HuggingFace repository. The dataset includes three classes (Good Fit, Potential Fit, No Fit), which we restructured into a binary classification task. Annotation labels reflect textual compatibility based on skills, responsibilities, and experience, ensuring task relevance.

It resulted in a moderately imbalanced dataset with approximately 66% positive and 34% negative examples. Labels were assigned based on semantic compatibility, including skill match, job responsibilities, and experience alignment. Our model achieved accuracy = 72%, precision = 70%, recall = 74%, F1-score = 72%, and Precision@10 = 75%, significantly outperforming both classical (TF-IDF + cosine similarity) and neural (Sentence-BERT without fine-tuning) baselines. These results validate the empirical effectiveness of our architecture for candidate ranking and selection. To justify the use of a complex Siamese architecture, the system was compared to two baselines: (1) a classical TF-IDF + cosine similarity method, and (2) a pretrained Sentence-BERT model without task-specific fine-tuning. The proposed model significantly outperformed both baselines across all evaluation metrics, confirming that its complexity translates to meaningful performance gains. A basic self-learning mechanism is implemented and functional. Recruiters can provide binary feedback (Fit / No Fit) for each recommended candidate, which is stored in a feedback table. This feedback can be used to retrain or fine-tune the model periodically, enabling adaptive behaviour over time. While initial retraining experiments were conducted offline, full automation and continuous integration of feedback into training pipelines remain a goal for future development. The system offers sub-5-second response times, integration with vector databases, and a web-based user interface. It is designed for use in HR departments, recruiting agencies, and employment platforms, with potential for broader commercial deployment and domain adaptation. We additionally implemented a feedback-driven retraining loop that enables future self-supervised adaptation. While UI and vector retrieval infrastructure were developed to support prototyping and deployment, the primary research innovation centres on the modelling framework, learning setup, and comparative evaluation methodology. This work contributes to the advancement of semantically-aware intelligent recruiting systems and offers a replicable baseline for future studies in neural recommendation for HR applications. The risks of algorithmic bias are emphasised separately: even in the absence of obvious demographic characteristics in the input data, the model can implicitly reproduce social or historical inequalities inherent in the data. In this regard, the study outlines areas for further development, in particular equity auditing, bias reduction techniques, and the integration of human validation in decision-making.

Index Terms: Candidate Ranking, Recommendation Systems, Precision@10, Top-N Recommendations, Relevance Evaluation, Information Retrieval, Transformers, Semantic Mapping, RoBERTa, SimSCE, MSELoss, Siamese Networks.

1. Introduction

In recent years, the increasing demand for efficient talent acquisition has driven the development of automated systems capable of recommending the most suitable candidates for job openings. These systems aim to assist recruiters in quickly identifying top talent from large applicant pools, reducing both time and effort. The primary objective of this work is to design and evaluate a recommendation approach that ranks candidates by relevance to a specific job description, with a particular focus on metrics such as Precision@10. By concentrating on the quality of the top-ranked recommendations, the goal is to ensure that the system delivers meaningful and actionable results to recruiters.

This study outlines the design choices, methodology, and evaluation framework adopted to measure the effectiveness of the proposed recommendation model in the context of real-world candidate selection scenarios.

Problem Statement. Despite the growing interest in automated candidate recommendation systems, ensuring that the most relevant candidates consistently appear among the top results remains a challenging task. Existing methods often suffer from low precision at the top ranks, which can lead recruiters to overlook qualified candidates or waste time reviewing irrelevant profiles. The challenge lies in designing models that can effectively capture the semantic similarity between job descriptions and candidate profiles, and optimise for evaluation metrics that reflect practical recruitment needs, such as Precision@10.

Motivation. In recruitment, the quality of the first few recommended candidates is often far more critical than the quality of the entire ranked list, as recruiters typically focus on reviewing only the top results. A system capable of accurately identifying and recommending the most relevant candidates among the first positions can significantly improve decision-making efficiency and reduce time-to-hire. Motivated by this, our work aims to develop and evaluate a recommendation approach that prioritises top-rank precision, ensuring that the candidates recruiters see first are those most likely to be a good fit.

Objectives

- To design and implement a recommendation model that ranks candidates by relevance to specific job descriptions.
- To evaluate the model using ranking-oriented metrics, with an emphasis on Precision@10, reflecting the practical needs of recruiters.
- To explore how semantic matching techniques (e.g., Siamese transformers) can improve top-N recommendation quality compared to baseline methods.

- To identify potential limitations and discuss how the model could be extended or improved for real-world deployment.

Contributions

- We propose and implement a candidate recommendation approach based on a Siamese transformer architecture to capture semantic similarity between candidate profiles and job descriptions.
- We conduct an empirical evaluation focusing specifically on top-rank precision, providing evidence of the model's practical value in recruitment scenarios.
- We compare the proposed model against baseline methods to highlight performance differences and demonstrate the effectiveness of semantic encoding for candidate-job matching.
- We provide a reproducible framework that can be adapted to similar recommendation tasks in human resources and related domains.

In today's world, the labour market is undergoing profound transformations due to the development of digital technologies, hybridisation of work formats and globalisation of business [1]. One of the key challenges for employers and HR specialists is the effective selection of personnel among an extensive array of candidates, often without the possibility of manual processing of all submitted resumes [2]. Despite the presence of numerous recruiting platforms, existing solutions are primarily focused on keywords rather than a deep semantic correspondence between the requirements of the vacancy and the candidate's experience. It leads to a high level of selection errors, the loss of potentially strong specialists or, conversely, the hiring of unsuitable persons. This problem is especially acute in the Ukrainian labour market, where there is a significant shortage of personnel in the fields of IT, engineering, medicine and other highly qualified industries. At the same time, the Ukrainian technology sector is actively developing, creating a need for competitive developments that can meet local business needs and adapt to the cultural and linguistic context. Against this background, innovative solutions based on modern advances in Data Science, in particular natural language processing (NLP) and deep learning, have incredibly high potential [3-5]. In particular, the use of transforming models in the tasks of semantic comparison of texts allows expanding the possibilities of automatic analysis of resumes and vacancies, improving the quality of candidate selection and reducing the time and resources spent on primary selection [5-8]. The development of an assistant recruiter based on such approaches is not only technically justified but also a socially significant step that can contribute to increasing the efficiency of the labour market in Ukraine. It also creates opportunities for the introduction of new business models in the field of HRTech and the development of national start-ups capable of reaching the global level. Thus, the chosen research topic is relevant from both a scientific and applied point of view and has the potential to have a tangible impact on the industry.

The scientific novelty of the developed solution lies in the combination of modern methods of natural language processing, vector search and deep learning architectures in the form of a holistic, automated recruitment system. This paper proposes an innovative approach to the presentation and comparison of textual descriptions of resumes and vacancies based on a common vector space formed using a Siamese transformer model trained using the corresponding loss function. This approach allows not only the classification of the "resume-vacancy" pairs but also the determination of the degree of their similarity on the scale without the need for rigid mark-up, which is of significant applied importance for the construction of recommended systems.

Unlike traditional approaches to automatic recruitment, which are based on rules, keywords or heuristic methods, this project implements model training based on the semantic comparison of paired examples. It made it possible to reduce the dependence of the system on superficial coincidences in texts and improve the model's ability to generalise to new, previously unseen examples. In addition, the principles of weak mark-up and positive-negative generation of pairs are used to form the training dataset, which minimises the need for a large amount of manual annotation and increases the scalability of the solution. Thus, the results of the work are not only of theoretical importance for the further development of the vector approach to text comparison but also open up practical opportunities for commercialisation and technological implementation in the field of modern recruitment.

The purpose of the study is to create an innovative prototype of the intelligent system "Recruiter-Assistant", capable of automatically performing semantic comparison of candidates' resumes with vacancy texts based on deep neural models and modern methods of natural language processing. Thus, to help HR departments, recruiting agencies, and technical managers find suitable candidates faster and more accurately, using deep semantic correspondence between the vacancy and the candidate's profile.

To achieve this goal, it is necessary to solve the following tasks:

- Analyse existing approaches to automated recruiting and identify their limitations.
- Generate a dataset of resume-vacancy pairs with relevant matching labels, including both positive and negative examples.
- Develop and implement a pre-processing pipeline for training models.
- Build an architecture of a transformer-based Siamese neural network (RoBERTa) using the corresponding loss function of the Triplet Loss type to train a common vector space.
- Train the model on the prepared data corpus and validate the results obtained.
- Evaluate the quality of the model by metrics of semantic proximity and candidate selection efficiency.

- Create and populate a vector database.
- Develop the application interface.

The object of the study is the process of digital transformation of recruitment, which includes automated analysis of candidates' resumes and job descriptions using machine learning methods and natural language processing. This process arises as a response to today's job market's need for a more accurate, fast, and scalable way to connect potential candidates with relevant jobs. The object is of particular importance in the context of a shortage of personnel, active migration of specialists and the growth of the remote labour market. The subject of the study is methods for building and optimising neural network models based on transformer architecture (in particular, Siamese models with a corresponding loss function), which allows encoding textual information from resumes and job descriptions in a common vector space in order to determine the degree of correspondence between them. The subject also considers the peculiarities of the formation of training samples, approaches to text pre-processing, and vectorisation, as well as the integration of the resulting model into a holistic product – a recruiter-assistant of a new generation.

Research Objectives. The primary objectives of this study are:

- To develop a semantic-based recommendation model capable of ranking candidates by relevance to job descriptions.
- To evaluate the model's effectiveness using top-N metrics, with a focus on Precision@10 as a measure of practical recruitment utility.
- To compare the proposed model against baseline approaches to demonstrate its strengths and limitations.

Novelty of the Study. The novelty of this work lies in applying a **Siamese transformer architecture** specifically tailored for ranking candidate profiles in recruitment scenarios. Unlike traditional keyword-based or shallow semantic methods, this approach captures more profound contextual similarities between job requirements and candidate experiences, leading to more relevant top-ranked recommendations.

Structure of the Paper. The remainder of this paper is organised as follows. Section 2 reviews related work on candidate recommendation and semantic ranking models. Section 3 presents the problem formulation and describes the proposed recommendation approach. Section 4 details the experimental setup and datasets used for evaluation. Section 5 discusses the results and their implications. Finally, Section 6 summarises the conclusions and outlines directions for future research.

2. Related Works

The modern labour market, especially in the field of technical specialities, is experiencing an unprecedented load due to the rapid development of technologies, globalisation and digitalisation of recruiting processes [6-8]. Companies are faced with the urgent need to quickly find qualified specialists while maintaining the quality of selection at a high level. At the same time, the number of candidates who respond to vacancies is often too large, which leads to an overload of HR departments with routine background checks and evaluations. Classic keyword-based approaches or manual filtering are ineffective in the face of large amounts of data and rapidly changing requirements. Another critical problem is the lack of contextual understanding of the correspondence between a candidate's resume and the job description. For example, two people may have the same experience but express it in different words, which makes it impossible to make an accurate comparison without deep semantic analysis. In addition, a large part of the resume may be written in atypical terms or describe experience in forms that do not coincide with the formal requirements of vacancies but are potentially relevant. Such situations often make it challenging to find a truly suitable candidate. The situation is even more aggravated in Ukrainian realities, particularly in the context of the outflow of personnel abroad, the lack of qualified specialists in certain regions, and the need to adapt to wartime conditions and remote work. It puts an additional burden on recruiting platforms and requires the implementation of flexible, smart, and adaptive solutions.

Recruitment is a process closely related to historical and social forms of prejudice, including gender discrimination, racial prejudice, ageism, and socioeconomic inequality. These biases are often hidden in historical data, especially if previous hiring decisions were made by people in a biased context. When machine learning models are trained on such data, algorithmic bias can manifest itself in several forms:

- Historical prejudice is the repetition of the inequalities of the past (for example, the priority of men's resumes).
- Bias – underrepresentation of certain groups in the data degrades the quality of predictions.
- Proxy bias – even without the direct presence of sensitive attributes (gender, ethnicity), other variables (for example, name, educational institution) can act as their indirect substitutes.

Potential consequences:

- Qualified candidates from marginalised groups may receive lower ratings.
- Models may conflict with the goals of inclusion and diversity in the company.

- There may be legal and reputational risks if the system demonstrates discriminatory results.

Mitigation strategies (implemented or planned):

- Feature filtering – demographic characteristics were not included in the training data.
- Fairness audit – in the future, it is planned to use metrics to check bias (for example, demographic parity, disparate impact).
- Data enrichment – you can increase the number of profiles from underrepresented groups.
- Human verification – the final decision is left to the recruiter, not the algorithm.
- Transparency – Explaining each recommendation increases trust and allows you to test the logic of the model.

As a result, fairness in recruiting should not be secondary; it should be the main criterion when creating and implementing AI models. The current study focuses on modelling and evaluation, but the subsequent phases of the project involve a deep integration of ethics criteria and anti-bias mechanisms.

Ethical Issues in Automated Recruiting:

- Bias on the basis of personality (gender, race, age, ethnicity). Resumes can contain direct or indirect signs:
    - Names that indicate gender or ethnicity.
    - Dates of birth, years of graduation (age).
    - Locations that may be associated with socioeconomic backgrounds or discriminatory stereotypes.

Risk – the model, even if it does not have explicit access to these features, can learn to "recognise" them through statistical relationships and replicate existing discriminatory patterns present in the training data.

- Bias in job descriptions. Job descriptions are often formulated using:
    - Gender-colored vocabulary (for example, "aggressive", "dominant", "nurturing").
    - Stereotypical requirements (for example, "must be energetic young professional").

Risk – the system may give preference to candidates whose resumes lexically or stylistically "coincide" with these descriptions, even if these requirements are not objectively necessary to perform the tasks.

- The problem of the "black box" (lack of transparency). Transformer-based semantic models are complex and poorly interpreted, making it impossible to explain:
    - Why a particular candidate is recognised as relevant/irrelevant;
    - Which fragments of the text played a decisive role in the decision?

Risk – the inability to explain or challenge an automatic decision is ethically unacceptable in the context of employment, especially in regulated industries.

The paper does not mention any analyses for model bias, in particular:

- No check for biased lexical patterns in the resume.
- It has not been investigated whether the model favours specific categories (e.g., men or young candidates);
- No procedures are proposed to detect or reduce bias, making the system ethically vulnerable to real implementation.

Recommended measures to reduce bias

- Diagnosis of bias:
    - Analyse the influence of sensitive traits (gender, age, ethnicity) on the classification result.
    - Apply Group Statistics: Compare Precision/Recall between groups (e.g. men/women).
    - Apply "bias testing" techniques, such as in NLP systems (for example, WEAT – Word Embedding Association Test).
- Masking sensitive signs:
    - Deletion or replacement of sensitive data (names, dates, addresses) during pre-processing of text.
    - Anonymisation of personal data in resumes.
- Rebalancing or redefining the sample:

- Ensuring equal representation of different social groups in the training sample.
- If necessary, unbalanced weight training: for example, an artificial increase in the weight of positive examples from underrepresented groups.

- Models with Fairness Control:

  - Integration of fairness-aware models that control discrimination (e.g. through adversarial debiasing).
  - Inclusion of interpreted rule-based overlays that allow "guaranteed" fairness to be imposed on an automatic decision.

- Transparency and Auditing:

  - Provision of decision-making logs.
  - Providing users (HR or candidates) with explanations why such a decision was made.
  - Possibility of feedback and appeal.

Automation of recruitment using transformer models and semantic comparison carries significant ethical risks associated with possible biases in input data. Without clear measures to reduce the discriminatory impact of characteristics such as gender, age or ethnicity, the system can inadvertently replicate or exacerbate structural inequalities. The current study does not analyse the presence or impact of such biases, which is a critical gap. For future work, it is necessary to introduce bias diagnostics, anonymisation of sensitive data, sample alignment, application of fairness-aware algorithms, and transparency of decision-making.

2.1. Classical Methods of Personnel Selection

Traditional approaches to recruitment, which were formed long before the advent of automated systems, are based mainly on manual analysis of candidates [1-2]. The main tools used in such methods are keywords, parameter filtering, and manual selection. Recruiters or HR managers search for resumes based on the presence of specific keywords or phrases, such as "Python", "Project management", "MBA", "3 years of experience", etc. This approach is widely used both in manual mode and in simple search engines. Many enterprise ATS (Applicant Tracking Systems) implement basic filtering by fields such as education, work experience, location, language proficiency, etc. One of the costliest processes is when each resume is reviewed manually, often under a tight deadline. The decision is made on the basis of a subjective assessment of compliance with the text of the resume and the job description. The disadvantages of classical methods are low scalability, inaccuracy due to variability of wording, dependence on the structure and format of the resume and subjectivity [9-12]:

- Manual analysis of resumes requires significant time and human resources. At least not too superficially. Processing even a few hundred resumes can take days or weeks. In cases of mass recruitment or the opening of several simultaneous positions, this approach becomes non-functional.
- Keywords may not cover the full palette of ways in which candidates describe their experiences. As a result, a relevant resume may not be included in the sample. An irrelevant one may be filtered since it depends very much on the recruiter himself, who may not fully understand what position a candidate is needed for and what will be included in his responsibilities, which skills are critically important and which are easily acquired.
- Many systems and recruiters work with template CVs, and if a candidate submits their resume in a non-standard format or creatively structured information, it may be ignored or misinterpreted.
- Classical methods largely depend on the individual experience, prejudices and professional intuition of the recruiter. It creates risks of discrimination, erroneous decisions or omission of non-standard qualified candidates.

2.2. Commercial Products

LinkedIn Recruiter is the largest platform for professional networking and recruiting. The leading technology used in the selection is keyword-based filtering, which takes into account location, experience, industry, etc. Recently, elements of AI recommendations have been added, but their basis still does not take into account the deep meaning of the text. Actually, the main problem is the search by keywords and filters, not semantic content. In addition, the search is carried out according to the candidate's profile, which should be filled with skills in a standardised format (selected from the list of proposed ones) and not submitted in any written form. Since the platform has the characteristics of a social network, recommendations quite often depend on the number of 'connections' of the candidate, which reduces the chances of getting an offer for people who have just registered.

SeekOut is a platform for "promising" recruitment focused on technical teams and start-ups. It has more AI features than LinkedIn, in particular, the construction of "talent markets", tools for analysing missing skills, and diversity filters. However, the model is also based mainly on attributive comparison. Although the platform is multifunctional and has its own database of candidates, this is a problem because the individual configuration of the system is not intuitive, and technical support assistance is often needed. In addition, it will take days for the system to adapt to the needs of the candidates and start selecting them.

Djini is a new player in the market that uses ML to select candidates based on career paths, soft skills, and non-obvious patterns. It is a Ukrainian platform focused only on the IT market and was created to provide a comfortable and anonymous job search. It is beneficial for candidates who want to change their place of work without advertising it. The system partially uses NLP approaches that take into account semantics as well as filters. The platform's focus is still more on candidates than on recruiters, and the scope of job search is customised for the IT field.

It is much more challenging to assess the effectiveness of the selection of vacancies and candidates since there is no access to the open source of these technical solutions, which makes it impossible to analyse approaches and ways to solve the problem [12-15].

2.3. Classic NLP Embedding Models

In the task of recruiting personnel based on automated analysis of vacancy texts and resumes, classical methods of Natural Language Processing (NLP) were traditionally used, including Bag of Words, TF-IDF, BM25, LSA, and Word2Vec. These approaches made it possible to carry out an initial assessment of similarity between documents using text vectorisation and calculation of similarity metrics (for example, cosine distance). Let us consider in more detail the potential of these methods for this problem [16-21].

Bag of Words (BoW) is one of the easiest ways to convert text to a numerical representation. The method forms a dictionary of all unique words in the corpus of documents. It presents each text as a vector, where each coordinate corresponds to the number of occurrences of the corresponding word. Word order and grammatical connections are ignored. For this task, BoW could be used as a basic approach in search engines when resumes and vacancies were presented as vectors of word frequency. For example, the system could count the number of common words between a resume and a job description and rank candidates based on this. This approach contains significant limitations, such as loss of context, ignoring semantic similarity, vector space, and sensitivity to wording:

- Depending on the location in the sentence, a word can not only have a different colour but also represent another part of speech or have an opposite meaning.
- A typical example and indicator of the versatility of language is the use of synonyms, which absolutely cannot be evaluated by this approach. During the selection of vacancies, this can have very negative consequences.
- For large dictionaries, BoW results in extremely sparse vectors, which degrades computational efficiency.
- Changing the order of words or using synonyms can significantly change the vector of the text.

TF-IDF (Term Frequency – Inverse Document Frequency) determines the importance of a word in a particular document relative to the total frequency in the entire collection of texts. It allows you to suppress frequently used words and emphasise weight on unique terms. TF-IDF can be used to calculate the similarities between a resume and a job by a set of keywords: for example, compare two texts as vectors in a term space and determine how close they are. The limitations that the approach contains include ignoring the context of words' semantic similarities, and there are also scaling problems:

- The model does not know in what sense the word is used because all words are considered in isolation. Respectively, even slight variations in wording lead to different vectors because the model does not see the linguistic affinity between them.
- Synonymous words are considered completely different because there is no mechanism for assessing their proximity.
- When adding new documents, each change in the dictionary requires a recalculation of the entire weighting matrix.

LSA (Latent Semantic Analysis) projects texts into a latent space using a matrix layout that displays hidden connections between words and documents. It can be used to identify hidden topics in vacancies and resumes in order to compare them. It contains the following restrictions:

- Requires a complete recalculation when updating the data (the singular decomposition of the matrix cannot be partially updated, so each addition of data requires a recalculation).
- Linear nature and insensitivity to language structures (the method is based on statistical relationships and is not able to take into account complex grammatical dependencies).
- Uninterpreted parameters (the number of measurements in the latent space does not have a clear value, and the selection of the value is purely empirical).

Word2Vec is an approach to building a vector representation of words, which is based on statistics of their contextual use. Unlike frequency models (like TF-IDF), Word2Vec learns the meaning of words based on their surroundings in the text. There are two main architectures: CBOW (Continuous Bag of Words), which predicts word by context, and Skip-gram, which predicts context by word. As a result of learning, each word is matched by a vector in a multidimensional space, where words that are close in meaning have a nearby location. In the problem of matching resumes and vacancies, Word2Vec allows you to compare documents not only by an exact match of words but also by semantically close concepts.

Usually, for this, the average vector of the document is calculated – that is, the vectors of all words in the text of a resume or vacancy are averaged, and these average vectors are compared by cosine distance. It allows you to partially take into account synonymy or related concepts even without repeating them exactly. Even this advanced approach contains certain limitations, such as ignoring word order and sentence structure, the equivalence of essential and unimportant words, and problems with words outside the dictionary:

- The model does not maintain a sequence or grammatical structure, so phrases with the exact words but in a different order will have the same representation.
- All words (regardless of their role in the context) have the same weight in the middle vector of the document, which can reduce the impact of key terms.
- The model works only with those words that were in the educational building; new or highly specialised terms are ignored or replaced with plugs, which is detrimental to quality.

2.4. Contextual Embedding

Modern NLP models, in particular transformers, have made it possible to move from static to contextual vector representations, where the meaning of a word depends on its environment. This greatly improved language comprehension at the level of phrases, sentences, and documents. Transformers are an architecture of neural networks that is based on the mechanism of self-attention, that is, the ability of each word to "carefully read" other words in a sentence and weigh their importance. It allows the model to capture complex dependencies between words, even if they are far apart. Compared to previous approaches (e.g., RNN or LSTM), transformers are much better at parallelising, learning faster, and scaling to large amounts of data.

BERT (Bidirectional Encoder Representations from Transformers) is one of the first deep transformer models that learns in two directions (left and right of the word) for a deeper understanding of the context. The model output is contextual vectors for each word, as well as the [CLS] token, which is often used as a representation of the entire sentence. In the problem to be solved, the model can be used to construct vectors for resumes and vacancies, which are then compared by cosine distance. It allows you to take into account context, synonymy, ambiguity and stylistic features. Certain limitations of the model are that it is not optimised for similarity problems and computational complexity. The basic BERT was not specifically trained on the issues of comparing the semantics of texts, so without additional tuning, the results may be irrelevant. Using a full-fledged BERT for each document in real-time requires significant resources, making it difficult to scale.

RoBERTa Robustly Optimised BERT Pretraining Approach is an improved version of BERT with optimised hyperparameters and a larger training body. Architecturally, it is the same, but more precise and stable. It can be applied to the task in the same way as BERT, but it will potentially give better results with preliminary training on HR domain data (resumes, vacancies, etc.). Among the limitations are the same problems as in BERT, in particular, the lack of specialisation for text comparison tasks and the need for significant computing resources.

2.5. Siamese Networks and Loss Functions

Analysing the problems of the methods mentioned above, there are two key ones: the lack of preservation of the semantic connection and the consideration of the context during embedding into a vector (transformers and the models built on their basis struggle with this to some extent) [3-5]. Another problem is that conditional training inputs come in resume-vacancy pairs, and the model needs to learn how to encode them into a single vector space [16-21]. One approach to solving this problem is the use of Siamese architectures. These models make it possible to train a common vector space, in which similar texts are placed closer to each other and different texts are placed farther apart, which significantly increases accuracy in matching problems, where it is essential to take into account both explicit and hidden semantic connections. A Siamese neural network consists of two (or more) identical branches with split weights that convert two input texts into vectors. These vectors are then compared on a specific metric (e.g., cosine distance), and the model is trained in a way that shortens the distance between similar pairs and increases it between different ones. In the case of a task, this means that the system learns to place relevant pairs closer in vector space and irrelevant pairs further away, providing efficient encoding. It is worth noting that Siamese networks are not an existing model but only an approach to training, where training data comes and goes through the embedding process in pairs. To achieve the desired properties of this vector space, Siamese architectures are combined with specialised loss functions that set the rules of learning: what "close" means, what "similarity" is, and what the distance between different examples should be. It is the choice of the loss function that determines how the model will respond to varying types of input pairs or triples and how it will learn to optimise its similarity space. Currently, the most popular functions used for encoding tasks in recommendation systems are the following:

- The CosineEmbeddingLoss function is used when you need to train the model to distinguish between similar and dissimilar pairs by cosine similarity. It works with labels 1 (similar pairs) and -1 (different pairs), trying to minimise the angular distance between similar vectors and maximise it for dissimilar ones. This loss function is interpreted and works well in similarity problems but may be less effective in the presence of "semi-similar" pairs or a wide range of relevance, which is often the case in the HR context.

- TripletMarginLoss operates on three examples: anchor, positive, and negative. It forces the model to minimise the distance between the anchor and the positive while maximising it to the negative. It forms a more structured vector space. TripletMarginLoss provides a clear hierarchy of semantic proximity, allowing for a more accurate ranking of results. The only problem at first glance is the need for careful formation of triples, which complicates the preparation of data.

2.6. Definition of Relevance in the Context of Research

Relevance in this study refers to the degree of semantic correspondence between the job description and the content of the candidate's resume, which the system determines based on their vector representation in the common feature space. It includes:

- Semantic similarity between the requirements of the vacancy and the candidate's experience (as opposed to a superficial match of keywords);
- The use of transformers (SimCSE-RoBERTa) takes into account the context of words.
- Training on positive and negative pairs (Good Fit / No Fit), which form the basis for the concept of relevance;
- Estimation due to the cosine proximity of the vectors.

Is relevance based on:

- Keyword crossovers? Nope. The authors explicitly indicate that classic keyword-based methods are ineffective and imperfect. The system is based on a deep semantic comparison, and not on a direct coincidence of words.
- Expert assessment in the field of HR? Partially. The training sample contains the labels Good Fit, Potential Fit, and No Fit, but it is not indicated whether HR experts gave these assessments. However, the system supports a feedback mechanism that allows recruiters to evaluate candidates, and these assessments are used to further train the model.
- Historical hiring data? Nope. There is no mention in the study of the use of historical data on the actual employment of candidates. The training data comes from open sources (HuggingFace, Kaggle) and contains the already labelled "resume-vacancy" pairs.

The lack of a clearly defined concept of "relevance" in the article is a methodological gap. After all, the subjective nature of conformity assessment in the HR field means that:

- Different companies or industries may have other ideas about relevance.
- Without a transparent criterion, the assessment of accuracy (72%) or Precision@10 (75%) may be unreliable or difficult to reproduce.

When deploying the system, you should:

- clearly formalise the criteria of relevance, preferably on the basis of professional expert assessments;
- indicate who exactly marked the data and what instructions they had;
- or recognise the subjectivity of relevance and add it as a limitation of research.

Within the framework of the study, relevance is defined as the semantic correspondence between the requirements of the vacancy and the candidate's professional profile, represented in the form of vectors in the common feature space constructed by the SimCSE-RoBERTa transformer model. A candidate is considered relevant if their resume:

- Contains skills, experience, education, or other characteristics that:
    - directly meet the requirements of the vacancy or
    - have functional or contextual similarity to them (taking into account synonymy, paraphrases, similar concepts);
- Demonstrates compliance with key aspects of the role, even in the absence of a literal coincidence of wording;
- Rated as "Good Fit" or "Potential Fit" in training data, simulating the expert opinion of HR professionals.

For automated relevance recognition, the system uses:

- vector representations of texts;
- cosine distance to calculate similarity;
- combined learning based on positive and negative pairs with labels that simulate expert assessments of relevance.

Despite the technical implementation of semantic search, relevance remains a subjective concept, since:

- The reference labels (Good Fit / No Fit / Potential Fit) in the dataset used are the result of an unknown or informalized labelling process, possibly inconsistent between experts.

- The concept of relevance can vary considerably depending on:
  - the specifics of the vacancy (for example, IT vs. humanitarian industries),
  - corporate culture of the company,
  - individual preferences of the recruiter or hiring manager.
- Lack of access to historical hiring data makes it impossible to check relevance based on the actual performance of candidates.

Because of this, the accuracy and Precision@10 measures given in the study should be interpreted as conditional, dependent on the original set of pairs and criteria applied during the mark-up of the data.

Recommendation systems for the selection of candidates are actively developing, combining methods of information search, machine learning and natural language processing.

Traditional methods. The first systems relied on **keywords and vector models** (TF-IDF), which made it possible to compare resume and vacancy texts by frequency characteristics [22-27]. However, these approaches did not take context and synonymy into account well, which led to poor quality recommendations under challenging cases.

Classical machine methods. Further development led to the use of classifiers (SVM, logistic regression) and ranking (BM25) [28]. Such methods made it possible to assess the relevance of candidates to vacancies, taking into account the statistical features of the texts, but remained sensitive to lexical discrepancies.

Semantic approaches using neural networks. With the advent of vector word models (Word2Vec, GloVe) and contextual models (BERT, RoBERTa), the ability of systems to capture deep semantics between job descriptions and candidate experience has significantly increased [29-31]. These models showed a marked increase in Precision and Recall scores, especially at the top positions of the list.

Siamese architectures. The modern direction is the use of Siamese transformers, where two identical encoders independently convert a vacancy and resume into vectors [32-34]. Next, similarity is calculated (for example, cosine), which allows you to effectively compare many "vacancy-candidate" pairs. This approach scales well and achieves better quality in top-N recommendation problems, in particular, Precision@10.

Metrics for quality assessment. The papers [35-36] emphasise the importance of focusing on the top-N metric (Precision@k, MAP@k, NDCG@k), because users usually consider only the first few results. In particular, Precision@10 reflects the proportion of relevant candidates the system recommends among the top ten.

Table 1. Comparison of candidate recommendation methods

Method	Semantic Understanding	Scalability	Top-N Metrics (e.g., Precision@10)	Context Awareness	Typical Limitations
TF-IDF / BM25	Low	High	Low	None	Ignores synonyms and context; relies on exact matching
Classical classifiers (e.g., SVM)	Low–Moderate	Moderate	Low–Moderate	Limited	Requires manual feature engineering; struggles with complex text
Word embeddings (Word2Vec, GloVe)	Moderate	High	Moderate	Limited	Captures local semantics but lacks deep contextual understanding
Contextual transformers (BERT)	High	Moderate	High	Strong	Computationally expensive; less suited for large-scale pairwise matching
Siamese transformers	High	High	High	Strong	Needs careful training; performance depends on data quality

Despite advances in semantic representation, many existing candidate recommendation systems still focus on global accuracy rather than the quality of top-ranked results. Traditional methods (e.g., TF-IDF, BM25) often fail to capture context or synonyms, leading to low precision among the first few recommendations. Classical machine learning approaches require manual feature engineering and may miss deeper textual semantics. Contextual models like BERT improve semantic matching but are computationally expensive for real-time or large-scale top-N ranking. Siamese transformer architectures have shown promise for efficient semantic similarity scoring, but research on their specific impact on recruitment scenarios, particularly measured by metrics like Precision@10, is still limited. This study aims to fill these gaps by:

- Applying a Siamese transformer model tailored for matching job descriptions and candidate profiles;
- Evaluating specifically on top-N metrics (e.g., Precision@10) to reflect real recruiter workflows;
- Demonstrating the model’s strengths and limitations compared to classical and contextual baselines.

3. Materials and Methods

3.1. Formation of the General Goal and Construction of the Tree of Goals

The general goal of the project is to simplify and, to some extent, automate recruitment processes, namely the selection of candidates for a vacancy, by introducing an intelligent recruiting assistant that uses modern methods of natural language processing, semantic search and machine learning. Such a system is designed to reduce the time that the recruiter spends on processing candidates' resumes while not worsening, but on the contrary, improving the quality of selection. A tree of goals is depicted, detailing the defined general goal – speeding up the selection of candidates for the vacancy by determining the semantic similarity of candidates and the vacancy while maintaining quality. This general goal requires the achievement of the following objectives:

- Development of an algorithm for selecting candidates based on semantic similarity, which in turn includes:
    - Collection and preparation/processing/labelling of data (in this case, resumes and vacancies);
    - Creating a model for embedding text into vectors while retaining all or most of the semantically meaningful information;
    - Create a vector database that will actually select candidates using vector similarity.
- There is a possibility of further improving the result since it is technically impossible to train the model on the entire available range of vacancies and resumes, given that formats and requirements change over time. This goal is achieved by collecting user ratings of the system's results, from which a dataset will be formed for additional training of the model based on recruiters' feedback.
- Creating a user-friendly interface is essential because no matter how powerful and innovative the algorithms used in the system are, if the user cannot interact with it, then the system is useless. Therefore, the interface should contain text prompts since convenience is quite subjective. Also, the ability to upload resumes in different formats will facilitate the process of interacting with the system.

3.2. Analysis of Alternative Ways to Achieve the General Goal

Let's also consider other algorithms/approaches for building a system that will select candidates for the vacancy. Since the essence of the existing idea is to embed vacancies/resumes, preserving semantic features and returning results based on the similarity of such vectors, the following solutions can be alternatives:

- Manual ranking of resumes with keyword filtering (rule-based). The idea of this approach is to highlight keywords, for example, hard and soft skills in resumes and vacancies, and compare matches – the more matches, the higher the score, and the candidate will rise higher in the ranking. The advantage of this approach is a more straightforward implementation since there is no need to learn or complete a large deep learning model (which requires a GPU). You can do without a vector database. In addition, the logic of candidate selection is as transparent as possible. There are similar words – high score, no–low. Among the disadvantages of the approach are the loss of context, dependence on format and wording, and the influence of synonyms and more complex grammatical constructions on the results, which are pretty significant.
- The idea of such a system has been superficially described many times before. It consists of using deep learning approaches and transformer models to encode text into vectors, which are then evaluated for similarity in a vector database. Preserve the semantic features of the text since it takes into account the context and not individual words, which makes the evaluation more complete and less dependent on the wording. The need for careful pre-processing of text and "extraction" of skills is also levelled since transformer models, on the contrary, work better with raw text. As for the disadvantages, there is a significant need for resources (GPUs) to train the model and dependence on training data (volume and quality).
- Aggregation of third-party services (LinkedIn/Djinni API). This approach involves the use of existing platforms to search for candidates to achieve a global goal. Logic. Among the disadvantages are the need for access to a third-party API, complete dependence on the stability of the selected system, where the logic will be implemented, and low controllability and transparency of the solution.

For a more visual comparison of alternatives, we will use the hierarchy analysis method. To do this, we will first determine the criteria by which we will compare alternative approaches.

- Selection accuracy – reflects how well the system is able to find a relevant candidate for a particular vacancy. This criterion should be taken into account since high accuracy increases the chances of successfully closing a vacancy and reduces the burden on recruiters, which is actually the formulation of the general goal of the project.
- Taking into account semantics – whether the system takes into account not just keywords but the meaningful, contextual meaning of words and phrases in resumes and vacancies, because, as mentioned earlier, in modern recruiting, semantic correspondence is more important than the superficial similarity of texts.

- Speed of implementation – estimates how much time and resources it takes to implement the system as an MVP.
- Flexibility and customisation – whether it is easy to modify the rules logic and adjust the model to the needs of a particular client or niche, because recruiting processes are often specific, the system must support adaptation to a particular area, company, or type of vacancy.
- Independence from third-party services, as long as the system does not depend on external APIs or platforms. This criterion is essential because dependency means the risk of changing access conditions, loss of control over data, and limited customisation, which will negatively affect the product.

Let's perform each of the choosing alternatives stages based on the analytic hierarchy process (AHP) (Tables 2-7).

Table 2. Matrix of pairwise comparisons of criteria

Criterion	Speed of implementation	Independence from third-party services	Flexibility and customisation	Taking into account semantics	Selection accuracy
Speed of implementation	1	2	3	4	5
Independence from third-party services	0.5	1	2	3	4
Flexibility and customisation	0.3	0.5	1	2	3
Taking into account semantics	0.25	0.3	0.5	1	2
Selection accuracy	0.2	0.25	0.3	0.5	1

Table 3. Matrix of pairwise comparisons for the selection of alternatives (criterion – Speed of implementation)

Speed of implementation	Manual ranking + filters	Transformers + vector database	Third-party APIs
Manual ranking + filters	1	0.5	4
Transformers + vector database	2	1	5
Third-party APIs	0.25	0.2	1

Table 4. Matrix of pairwise comparisons for the selection of alternatives (criterion – Independence from third-party services)

Independence from third-party services	Manual ranking + filters	Transformers + vector database	Third-party APIs
Manual ranking + filters	1	1	5
Transformers + vector database	1	1	5
Third-party APIs	0.2	0.2	1

Table 5. Matrix of pairwise comparisons for choosing alternatives (criterion – Flexibility and customisation)

Flexibility and customisation	Manual ranking + filters	Transformers + vector database	Third-party APIs
Manual ranking + filters	1	2	0.5
Transformers + vector database	0.5	1	0.3
Third-party APIs	2	3	1

Table 6. Matrix of pairwise comparisons for the selection of alternatives (criterion –Taking into account semantics)

Taking into account semantics	Manual ranking + filters	Transformers + vector database	Third-party APIs
Manual ranking + filters	1	4	3
Transformers + vector database	0.25	1	0.5
Third-party APIs	0.3	2	1

Table 7. Matrix of pairwise comparisons for the selection of alternatives (criterion –Accuracy of selection)

Selection accuracy	Manual ranking + filters	Transformers + vector database	Third-party APIs
Manual ranking + filters	1	4	3
Transformers + vector database	0.25	1	0.5
Third-party APIs	0.3	2	1

After calculating the vectors of local priorities and performing other intermediate calculations, the final vector of priorities of the alternatives was obtained (Table 8).

Table 8. Final comparison table of alternatives

Alternative	Priority
Manual ranking + filters	0.18376575
Transformers + vector database	0.515402953
Recruiting Platforms API	0.300831297

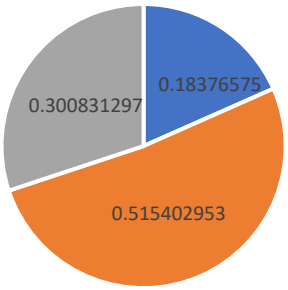

Fig.1. Visualisation of the priority of the considered alternatives, where blue is a problem with filters, semi-orange is transformers with vector databases, and grey is third-party APIs

3.3. Building the Architecture of the System

Let's describe the relationships between users (actors) and the system in order to understand what functionality is available to each of the actors. In this system (recruiter-assistant), the actors are the recruiter himself, as well as the system administrator (DL-engineer). The recruiter uses the system for its primary purpose. At the same time, the administrator has access to the system logic and can edit the database and the model itself, which makes the process of additional training based on user feedback real. The primary methods available to the recruiter are:

- Insert a vacancy (the text of the vacancy and/or a link to the vacancy);
- Choose the number of candidates he wants to receive.
- Get a list of candidates and their suitability (similarity);
- Assess the suitability of candidates for the vacancy and save the score in the database.

The main processes available to the system administrator are:

- Edit the vector database (including updating the candidate table);
- Update the model, including training the model on the data from the rating table.

A class diagram displays the structure of a system through its classes, their methods, and their relationships. The diagram shown in Figure 2 is somewhat abstract because it does not display all the details of the code but summarises them to form a more complete picture of the system's architecture. So, the system consists of the following classes: model class, word processing, vector database, user interface, and administrator services. As you can see, the administrator class does not have any connections to the interface class since it interacts with the model and database directly and not through the interface.

A sequence diagram (Figure 3) is a type of behavioural diagram used to visualise interactions between objects within a particular scenario or process. It focuses on the order of method calls, message transmission, and the life cycle of objects over time. This diagram helps to display the modules of the system and, most importantly, the method and sequence of communication between them. We describe the set of states of the system and the cause-and-effect relationships of transitions between them, which is usually depicted in the form of a coherent graph. In the context of this system, the node graph displays all possible states of the recruiter assistant during his interaction with the user (recruiter).

Step 1. Waiting for the text of the vacancy. If the Input type 'from URL' is selected, go to step 2. If the text is entered, the get embedding button is pressed, then go to stage 4.

Step 2. Waiting for a link to a vacancy. If the link is added, the get embedding button is clicked, then go to step 3.

Step 3. The text received after parsing the web page is saved in a variable.

Step 4. The text processed by the pre-processor is stored in a variable.

Step 5. Received text embedding and saved to a variable.

Step 6. The vacancy and its embedding are saved in the database. If k is set, the Find candidates button is pressed, and then proceed to step 7.

Step 7. The similarity of vectors and candidates is calculated and stored in a variable.

Step 8. The top k candidates and their similarities are displayed. If the results are evaluated (fit/no fit) and the Save Feedback button is clicked, then proceed to step 9.

Step 9. The score is saved to the database, and a message about this is displayed in the test field.

This series of diagrams is completed by the one that displays the physical location of software components on the hardware (Figure 4). It shows the nodes (servers, clients, devices) on which software artefacts (modules, services, databases, etc.) are deployed and the relationships between them.

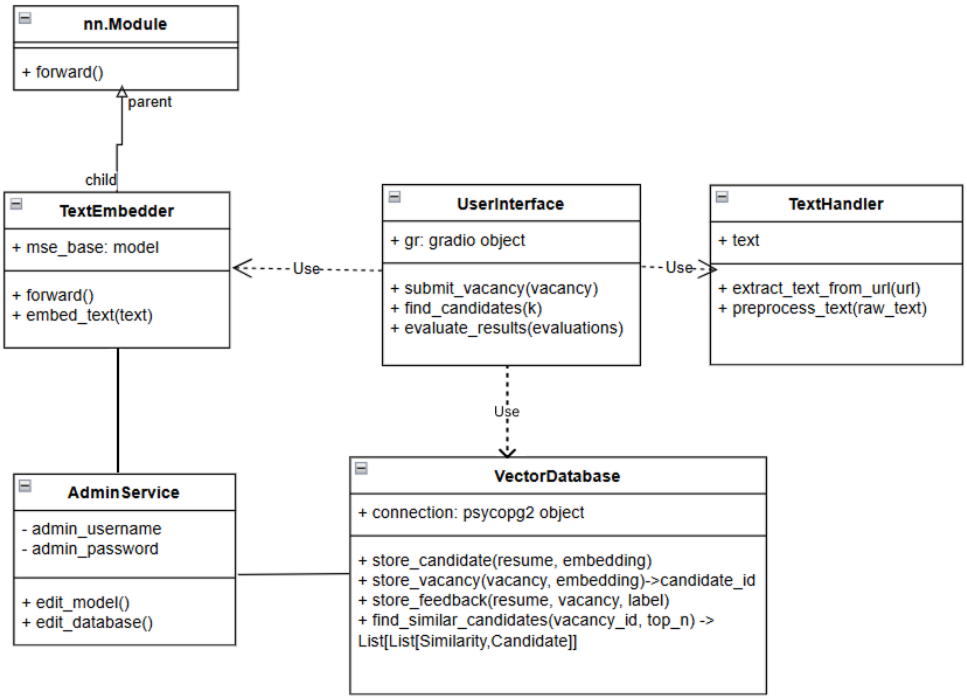

Fig.2. Diagram of system classes

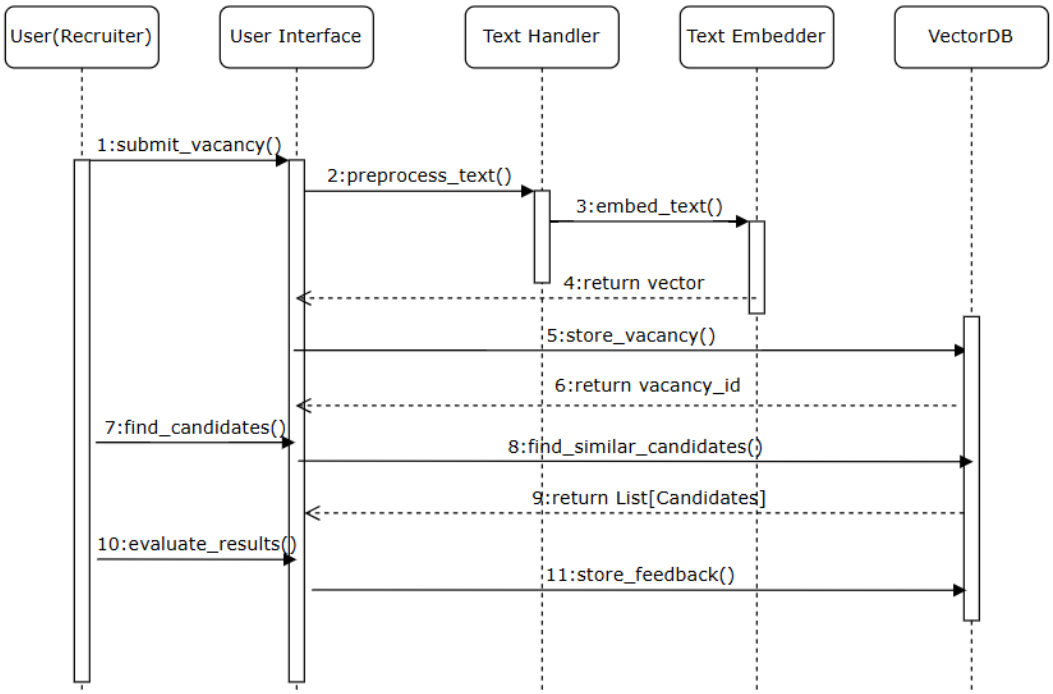

Fig.3. System sequence diagram

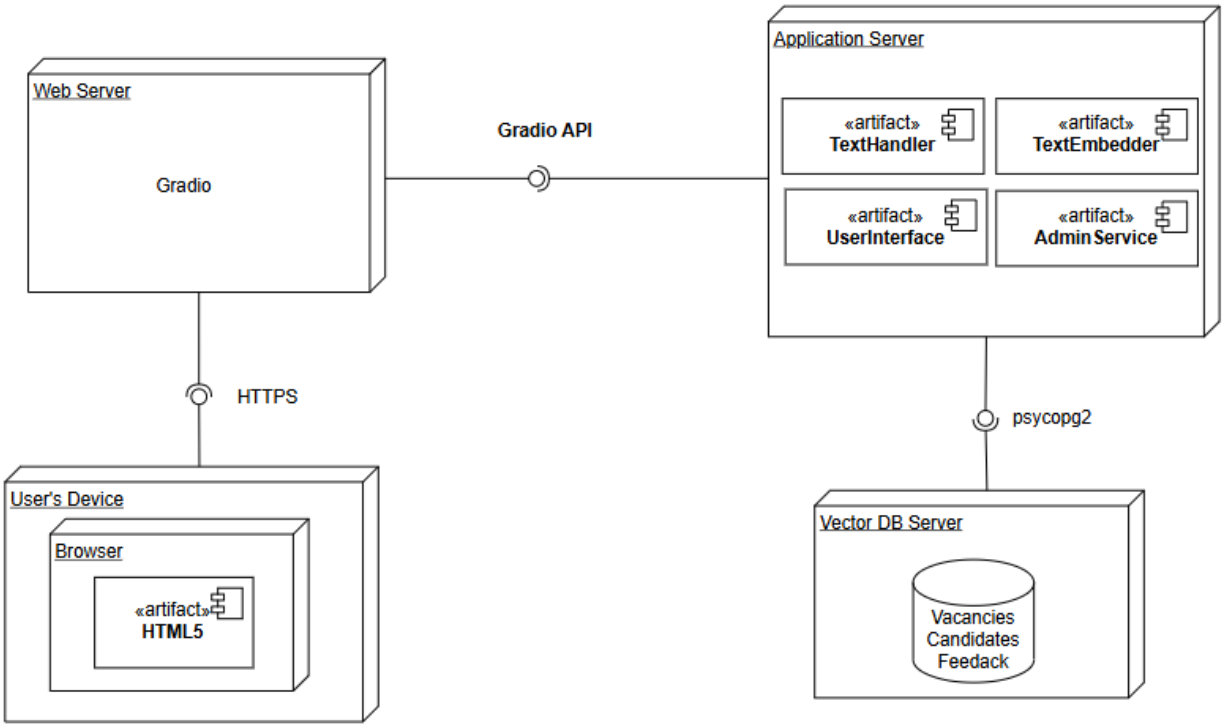

Fig.4. System deployment diagram

3.4. Purpose and Scope of the System

The system is initially designed to automate and speed up the selection processes for the relevant vacancy. It allows you to select the number of candidates specified by the user for the vacancy entered by them. Further training on the model is also envisaged. The user will be able to leave an assessment of the relevance of each returned candidate for a given vacancy, thus creating data for additional training of the model. Thanks to the use of transformer models and modern loss functions, the system achieves high sensitivity to semantic correspondence. It allows it to handle more flexible or non-obvious job descriptions where keywords may differ significantly, but the content remains compatible. As a result, the system reduces the human factor, increases the speed of hiring, and allows HR specialists to focus on more strategic aspects of recruitment. The application of the system is envisaged in the HR field, in particular:

- HR departments of companies that have a large flow of applications want to reduce the time needed to find suitable candidates.
- Recruiting agencies that provide recruitment services for multiple clients and seek to automate routine tasks.
- Freelance interviewers or HR consultants who are looking for a tool for pre-screening candidates;
- Online employment platforms that want to improve the algorithms for recommending vacancies/candidates.

Although the idea can be applied to any other recommendation system.

3.5. Justification for the Development and Implementation of the System, Description of Business Processes

The need to develop such a system is based on the urgent needs of the labour market, modern challenges in the field of recruitment, and limitations of traditional approaches to the selection of candidates.

Firstly, the volume of resumes that recruiters process on a daily basis far exceeds the possibilities of manual verification. Large companies receive hundreds, and sometimes thousands, of applications for one position. Manual assessment of the compliance of resumes and vacancies is not only a time-consuming process but also one that can be subjective. An algorithmic approach allows you to reduce the burden on HR specialists by weeding out irrelevant applications at the stage of preliminary screening.

Second, traditional keyword search filters are often ineffective. They do not take into account semantic similarity, synonymy or context. For example, a candidate with "experience in REST API development" may not be included in the sample if the job description mentions only "backend service creation". The use of deep learning language models allows the meaning of phrases to be taken into account, not just their superficial form, providing a much more accurate assessment of compliance.

The third important factor is the reduction in hiring time and cost. Quickly finding relevant candidates or vacancies reduces time-to-hire and allows companies to quickly fill critical positions. At the same time, automation of processes enables you to reduce the cost of finding personnel, especially in conditions of high competition in the market.

From a technical point of view, the rationale for implementation is based on the fact that modern neural network

architectures allow you to effectively train models on open or internal datasets, adapting them to the specifics of the business. The system can be easily scaled, integrated into HRM platforms and employment portals, or work as a standalone solution for recruiters.

A BPMN diagram has been developed to provide a more visual representation of the system and its future functions. It describes the previously mentioned functionality of the system, helps to clearly delineate the 'areas of responsibility' and covers the entire life cycle of interaction between the recruiter, the system, and the vector base – from the moment of entering a vacancy to collecting feedback for additional training of the model.

Step 1. Elaboration of processes from recruiters:
Step 1.1. Choose an input format.
Step 1.2. Insert a vacancy.
Step 1.3. Click the Get Embedding button.
Step 2. Elaboration of system processes:
Step 2.1. Checking the input format. If it is text (not URL), then go to step 2.3; otherwise, go to step 2.2.
Step 2.2. Scribing a web page;
Step 2.3. Save the resulting vacancy text to a variable.
Step 2.4. Process the text of the vacancy with a pre-processor.
Step 2.5. Apply a model for job embedding.
Step 2.6. Send a request to the database for addition.
Step 3. Elaboration of vector database processes:
Step 3.1. Save the vacancy and embed it in the Vacancy table.
Step 3.2. Return the ID of the saved vacancy.
Step 4. Elaboration of system processes:
Step 4.1. Display ID;
Step 4.2. Display a message about the completion of the vacancy embedding.
Step 5. Elaboration of processes from recruiters:
Step 5.1. Select the number of candidates (k);
Step 5.2. Click the Find Candidates button.
Step 6. Elaboration of system processes: send a request to the database for the selection of candidates.
Step 7. Elaboration of vector database processes:
Step 7.1. Calculate the similarity of vacancy vectors and candidates.
Step 7.2. Return the top k candidates and their similarity.
Step 8. Elaboration of system processes: display candidates and similarities.
Step 9. Elaboration of processes from recruiters:
Step 9.1. Assess the suitability of the selected candidates (Fit/No Fit):
Step 9.2. Click the Save Feedback button.
Step 10. Processing of system processes: send a request to the database for addition.
Step 11. Working out the processes of the vector database: save the scores to the Feedback table;
Step 12. Processing system processes: display a message about the save status.

3.6. Description of System Requirements

Based on the analysis and built diagrams of the system, we will describe the requirements.

General requirements for the system:

- Type of tool – a web interface for selecting relevant candidates by the text of the vacancy or a link to it;
- Interoperability:
  - The system should be able to read information from resumes available on web resources at the link using parsing.
  - Ability to accept resumes also in text format.
- Mobility:
  - Full accessibility from any device through a browser.
  - No need to install additional software.
- User Experience – Intuitive UI with hints and the ability to:
  - Enter the text of the vacancy manually.
  - Add links to an external vacancy.

FURPS+ Quality Requirements:

- Functionality:
  - Semantic search for candidates – taking into account the content of the vacancy and implicit connections in the text (synonyms, paraphrasing, professional vocabulary).
  - Generation of top-N recommendations – automatic output of the most relevant candidates by vector similarity.
  - Explainability – interpretation of why exactly these candidates are suitable (highlighting key coincidences, semantic correspondences).
- Usability:
  - Simple, straightforward UI/UX without the need for training;
  - Visual indication of relevance (for example, in percentages or colours);
- Performance:
  - Response time to the search for candidates: up to 60 seconds.
  - Job encoding time: less than 30 seconds;
  - Support for scalable processing: through the use of a vector database.
- Reliability:
  - Redundancy, fault tolerance;
  - Automatic model recovery;
- Supportability:
  - CI/CD pipeline for updating models;
  - Automated monitoring of model accuracy in the production environment;
- Additional requirements – availability of Active Learning:
  - Collecting feedback from the recruiter on the relevance of the results.
  - Periodic additional training of the model on this feedback.

3.7. Goals and Effects of their Implementation

The primary focus is on developing a model for selecting candidates for the proposed vacancy based on semantic similarity. Let's highlight more specific goals:

- Development and implementation of the core of the semantic search system:
  - Selection and training of an embedding model for coding resumes and job descriptions, which will allow you to get as much semantic context as possible from resumes and vacancies.
  - Construction of the process of indexing vectors in a vector database for effective search in large data arrays;
  - Implementation of a candidate search system based on vector proximity.
    - ✧ Automation of recruiting processes with the help of AI recommendations. The system should not only return relevant candidates but also:
    - ✧ Assess the correspondence between candidates and the vacancy (scoring).
    - ✧ Form an explanation of selection based on the attention/importance of individual fragments (highlighting key correspondences between the resume and the vacancy);
- Building a scalable and reliable technical architecture – using asynchronous microservices
- Integration with external HR systems and data sources – the ability to automatically extract vacancies from sites using NLP pre-processing.
- Building a self-learning system with user feedback – introduce feedback mechanisms based on user choice/refusal and additional training.

Table 9. Effects of the implementation of the selected objectives

Target	Expected implementation effect
Implement effective semantic search based on the embedding model	– A significant increase in the relevance of the selection of candidates compared to classic keyword-based systems. – Reducing the number of missed relevant candidates – Search versatility for different markets
Using a vector base to store embedded vectors	– Instant search through an extensive database of resumes, thanks to vector indexing and approximate search algorithms. – Flexibility to scale the system according to the size of the client, from start-ups to large recruitment agencies.
Add AI recommendations and selection explanations	– Automation of routine sorting of resumes, saving time for HR specialists (up to 60-70% of efforts at the first stage). – Increase trust in the system through clear explanations (highlighting why this particular candidate is at the top). – Reduction of time-to-hire due to better prioritisation of candidates.
Build a scalable architecture.	Reliability and continuity of the system, even under load
Integration with external sources	– Reduction of manual data entry and automatic reading from web resources will reduce the burden on HR. – Improving data quality through NLP pre-processing (normalisation of experience, skills, etc.).
Feedback-based self-learning system	– Improving the accuracy of recommendations over time because the system becomes smarter thanks to the interactive learning cycle. – Individualise recruiting logic for a company or even a recruiter.

The effectiveness of the recruitment assistant is based on a noticeable improvement in the quality of candidate selection and a significant impact on the recruitment processes as a whole.

First of all, the implementation of semantic search based on the embedding model demonstrates the high relevance of the results. The system is able to effectively take into account the context, synonymy, and hidden connections between the candidate's experience and the requirements of the vacancy, which is significantly superior to traditional keyword-based approaches. This results in fewer missed relevant candidates and improved search accuracy. The vector database, which supports proximity search, provides stable performance even with large amounts of data and allows you to scale the solution depending on the needs of the business.

On the user interaction side, AI recommendations with explanations (highlighting relevant fragments) make the system more transparent for HR specialists. It helps to increase trust in the system and allows you to save significant time sorting resumes. In combination with additional training mechanisms that take into account user feedback, the system gradually adapts to the needs of the market.

Taken together, this method of recruiting not only solves the problem of recruiting more accurately and quickly than most available analogues but also demonstrates a high level of adaptability, scalability, and automation.

3.8. Determination of Methods for Solving the Problem

As part of the development of an intelligent system for the selection of relevant candidates according to the job description, available approaches to the presentation of knowledge, logical inference mechanisms and decision-making algorithms have been analysed. The system is focused on processing unstructured text data (resumes and job descriptions), so the focus is on modern natural language processing (NLP) and semantic comparison methods.

Input data – sources of receipt and pre-processing. Training data has a significant impact on the efficiency and accuracy of the model and, in fact, the systems. For the problem to be solved, a dataset is needed, which will contain the texts of vacancies, the texts of candidates' resumes and, preferably, in some way, the relationship between them. Resources for searching datasets, and not only those related to this domain, are entire platforms, such as HuggingFace or Kaggle. An additional option for obtaining data is scribing HR web resources and independently forming a dataset. This approach is unproclaimed data, and in order to train the model on such data, it is necessary either to label the data independently (which requires a lot of time and also adds a significant subjective assessment to the data and, accordingly, the model) or to create an intermediate model or use an existing one to build certain relationships between the data (which also requires time, additional computational resources and does not guarantee high efficiency of such data in the future). However, to some extent, this method will be used to form a dataset for additional model training (user feedback + scribing + stored data). Therefore, online resources with datasets remain the main option. If necessary, the data found can be contacted or modified in a certain way to achieve the set goals.

Pre-processing of data does not require much effort since the use of transformer-based models for text encoding was previously agreed upon. Such models do not require careful pre-processing. On the contrary – by "purifying" data in ways that are typical for approaches that are not semantically sensitive (TF-IDF, Bag of Words, etc.), the results of the transformer model deteriorate because it loses crucial context. In this way, superficial pre-processing will be applied – lowercase reduction, removal of HTML tags (may remain after scribing), and possibly removal of duplicate characters and some punctuation characters. The dataset will need additional manipulations depending on the selected loss function, since it requires a different pair structure. In general, it can only be positive couples; the so-called triplets are anchor, positive pair, and sharply negative pair; anchor, positive pair, list of negative pairs, etc.

Methods and means of presenting knowledge, model and function of losses. The presentation of knowledge in the system is implemented through the vector representation of texts using semantic embedding. Based on the approaches previously analysed to solve the problem of similarity of texts, the effectiveness of transformer models (in particular based on BERT) is highlighted, which allows you to select and store semantic features of the text and generalise them. Siamese networks are also highlighted as an approach to encoding (processing) a pair of texts, which is ideal for the resume-vacancy option and will help encode texts into the same dimension. It provides a representation of the semantic content of documents in a vector space, allowing comparisons to be made based on similarities.

Knowledge about the relationship between a resume and a vacancy is presented not through formal rules but through training the model on a large dataset of pairs (resume, vacancy) with binary conformity assessments. Thus, the model learns the latent patterns of correspondence between the job description and the candidate's profile.

A large share of the system's success falls on the efficiency and correctness of the embedding model, which, as previously defined, will be built on the basis of the transformer's BERT. The question arises as to which of the available models to use. At the very least, there are conventional and sentence transformers based on BERT and, accordingly, their modifications. Since there is no unambiguously good or bad model for a particular task, the list of selected transformers will be tested on the generated test dataset to determine the most effective one. Among the many existing models, the following are chosen: BERT, RoBERTa and SimCSE-RoBERTa (sentence transformer).

The accuracy of the model, especially in embedding problems, is significantly influenced by loss functions, and this problem is no exception. Again, there is no predetermined good solution. It will be made as a result of testing the loss function on a set with previously defined patterns. The following loss functions have been selected for consideration: TripletMarginLoss, CosineEmbeddingLoss, MSEloss, and InfoNCE.

Mechanisms of logical inference. Unlike classical expert systems, which use a production model of knowledge with strict rules ("if-then"), the proposed system uses implicit logical inference through a neural network. After encoding the text into vectors, the cosine similarity is calculated, which acts as a degree of correspondence between the vacancy and the resume. Thus, the logical output is replaced by a vector operation, which scales much better and works better with fuzzy or partial coincidences.

Decision-making algorithms. The main decision-making algorithm is the semantic comparison of vectors and the ranking of candidates based on similarity. In particular:

- The incoming vacancy is encoded in a vector.
- All Knowledge Base candidates are also represented vectorially.
- The similarity between the vacancy and each resume is calculated.
- The results are sorted, and the top N candidates with the highest scores are formed.

For large-scale processing, a vector database is used, which allows you to search for the nearest neighbours in high-dimensional space in milliseconds.

The only fragment of the decision-making algorithm that needs to be discussed is the choice of evaluating the similarity of vectors. The distance between them mostly calculates the similarity of vectors, but there are many methods for finding this distance. We will choose among those supported by vector databases:

- L2 distance – calculates the "straight" distance between two points in a vector space. Suitable for non-sparse vectors but sensitive to the scale of values.
- (Negative) inner product – scalar product. The larger the inner product, the closer the vectors, so the negative value is convenient for optimisation. It is well suited for cases where not only directions but also magnitudes are essential.
- Cosine distance – complementary to cosine similarity – focuses on the angle between vectors, ignoring their length. It is the most common choice for problems with the semantic similarity of texts.
- L1 distance is the sum of the moduli of the differences between the components. It is emission-resistant but rarely used for semantic search.
- Hamming distance is the number of positions in which the bit values of two vectors differ. It applies only to binary vectors.
- Jaccard distance – evaluates the ratio of the intersection to the union of sets. Also, it is only suitable for binary vectors.

Some methods are suitable only for binary vectors, and some are rarely applied to such problems, so the choice will fall on the cosine distance since it is a standard solution. Still, in general, the method of calculating the distance should not affect the result.

During the analysis of methods for solving the problem, options are considered, and some decisions are made. Let's summarise this information and consolidate the information received earlier in tabular form.

Table 10. Defined methods

Task	Solution method
Search by semantic similarity	Text Embedding + Vector Similarity Search (previously defined)
Text Embedding	o BoW o TF-IDF o Contextual embedding ✓ **Siamese Networks + BERT-Based Contextual Embedding**
Transformer model	It will be selected by testing among the following: o BERT o RoBERTa o SimCSE-RoBERTa
Loss function	It will be selected by testing among the following: o TripletMarginLoss o CosineEmbeddingLoss o MSEloss o InfoNCE
Method for calculating similarity	o L2 distance o (Negative) inner product ✓ **Cosine distance** o L1 distance o Hamming distance o Jaccard distance
Dataset formation	o Independently, with the help of scribing and labelling ✓ **Existing datasets (Kaggle, HuggingFace) + modification**
Pre-processing of data	✓ **Lowercase Reduction** o Removing word stops o Lemmatisation/Stemming o POS/NER ✓ **Remove HTML tags** ✓ **Remove specific punctuation characters** ✓ **Removal/Replacement with Link Tags** o Removal/Replacement with Number Tags

3.9. Determination of Means of Solving the Problem

The development of an intelligent information system for the selection of candidates according to the content of vacancies requires the involvement of a wide range of software, infrastructure and auxiliary tools. This subsection provides a comparative overview of the available tools that can be used to solve the problem, which includes the stages of machine learning, API implementation, visualisation, data storage, and vector search.

Encoding Model Development Tools

The primary implementation language is Python, which is the leading language in the field of Data Science and Machine Learning. Python has a large number of libraries for natural language processing (NLP), including transformers, torch, scikit-learn, and nltk, which are used for data pre-processing, vector representation, and similarity metrics. The model is a key element of the system, and its development requires not only access to specific libraries but also significant computing resources. Among the possible development environments, the following options are considered:

- Google Colab is a convenient platform for solving machine learning and deep learning problems. There is a free version with a limited number of resources. It contains many built-in libraries and allows you to download additional ones. The main disadvantage is the format and volume of free computing resources since Google does not publish a clear number of GPUs that are provided and the duration of its updates. Rules for obtaining resources are minimal.
- Kaggle Notebooks is a user-friendly platform that is very similar to Colab. The procedure for providing resources is much more transparent: the GPU is limited in the time of use - it is 30 hours per week. If necessary, you can additionally use TPU, but this resource is provided on a first-come, first-served basis. However, since this is an add-on over Kaggle, it is easy to interact with Kaggle datasets and models, and you can generate and upload your own.
- Local development – allows complete control over the environment's custom settings depending on the choice of the environment, but requires significant local computing resources to train the model, which can be a problem.

For training transformer models and generating vector representations of texts, the following basic frameworks are considered:

- PyTorch is flexible and convenient for research and has good integration with HuggingFace Transformers, but the process of fine-tuning/transferring learning models is somewhat more complicated.
- TensorFlow / Keras is an alternative with a powerful, production-ready API, but it is less flexible when configuring pre-trained models from third-party resources.

- JAX/Flax is the newest alternative with a focus on performance, but it is less supported in the NLP ecosystem.

API Implementation Tools

There are several approaches to building an interface between the model and external services. Among them is creating your front-end using HTML/CSS/JavaScript, which would give you complete control over UI/UX but would require much more time for development and maintenance. Another use case is FastAPI, which, in combination with Postman or Swagger UI, can be used to build a REST API and test interoperability. However, for the task of quickly creating an interactive interface for accessing the ML model without unnecessary effort, the Gradio framework was chosen. Gradio provides ready-made components for data entry (text, links, PDFs), allows you to quickly visualise results, supports user feedback processing, and easily integrates with PyTorch models and machine/deep learning models in general. Thus, Gradio will significantly speed up the deployment of the MVP version of the system.

Storage and vector search systems

In modern candidate selection systems, it is essential not only to store textual information but also to ensure an effective search in vector space. Several solutions were considered to implement the repository of vector representations and implement similarity search. In particular, FAISS from Facebook AI is a separate in-memory system with high performance, and Milvus is a full-fledged vector database with support for scaling and an API. However, these solutions require separate hosting and more complex integration with the overall system.

Due to the need to store additional metadata (vacancy, text, link) along with the vector, PostgreSQL with the pgvector extension was chosen. This solution allows you to combine relational storage and vector search in a single DBMS, provides support for basic metrics (cosine similarity, L2 distance, etc.) and scales well in cloud environments. PostgreSQL is open-source software that removes licensing restrictions and allows you to use it in a start-up environment at no additional cost.

System Deployment Tools

The database is deployed in a Docker container, which allows you to standardise the development and testing environment. Containerization makes it easy to scale components, provides convenient integration into CI/CD pipelines, facilitates system upgrades, and simplifies its deployment both on-premises and in the cloud. Alternatively, virtual machines or cloud-based PaaS solutions (e.g. Heroku, Render, Railway) can be used. Still, Docker provides more flexibility and control, which is critical during the prototyping and experimentation phase.

The Gradio-enabled interface itself will be deployed locally in the PyCharm environment in Jupiter Notebook format. PyCharm has many application environments, and it also integrates with other resources. In this case, the key was the possibility of convenient integration with PostgreSQL. Let's summarise the results of the unit in tabular form for clarity.

Table 11. Defined means

Field	Remedy
Developing an Encoding Model	
Programming language	Python
Development Center	o Google Colab ✓ **Kaggle Notebook** o Locally
Framework	✓ **PyTorch** o TensorFlow / Keras o JAX / Flax
API Implementation Tools	
o Creating your own frontend + FastAPI ✓ **Gradio**	
Storage and vector search systems	
o FAISS o Milvus ✓ **PostgreSQL with pgvector extension**	
System Deployment Tools	
Docker for DB and PyCharm for integration with Gradio	

3.10. New Aspects of Model Architecture and Methodology

Despite the fact that the study is accompanied by a high-quality implementation of the system interface and infrastructure (Gradio, PostgreSQL with pgvector, Docker), its principal scientific value lies in the improved architecture of semantic correspondence between the resume and the vacancy. The following are the key groundbreaking components:

- Semantic model taking into account weakly marked pairs:

- The use of the Siamese architecture with two branches encoding the text of the vacancy and resume into a common vector space.
- The model trains on weakly labelled data (No Fit / Potential Fit / Good Fit), while:
  - ✧ The combination of positive classes (Good + Potential Fit) was carried out to increase the inclusiveness of the model.
  - ✧ An adaptive loss function was built (TripletMarginLoss, CosineEmbeddingLoss, MSELoss, and InfoNCE were tested), which allows you to control the "voltage" between classes without strict threshold control.
- Scientific novelty – flexible adaptation to fuzzy classes in compliance problems without the need for exact "ground truth", which allows the approach to be applied in non-standardised HR contexts.

- Using SimCSE-RoBERTa as an optimal sentence encoder:
  - The SimCSE (Simple Contrastive Learning of Sentence Embeddings) model is used in conjunction with the RoBERTa architecture, which provides a deep contextual representation of documents.
  - Advantages over conventional sentence-transformers:
    - ✧ The model is trained in the style of contrastive learning, so it better captures semantic differences between close texts.
    - ✧ A CLS token with L2 normalisation is used, which stabilises the calculation of the cosine distance between vectors and reduces the impact of document length.

Scientific novelty – the effectiveness of SimCSE-RoBERTa + MSELoss in HR recruiting tasks, which has rarely been evaluated in previous studies, has been empirically proven.

- Hybrid strategy for building a training dataset:
  - A hybrid approach to the formation of educational examples is proposed:
    - ✧ Use of annotations from a public dataset (HuggingFace);
    - ✧ Merging with Skyring data (for instance, with LiveCareer);
    - ✧ Use of pseudo-negative generation (anchor-positive-negative pairs) to work with Triplet Loss.
  - Scientific novelty – the expansion of the learning space through the controlled creation of "half-truth" triples allows the model to learn on ambiguous pairs, which is characteristic of real HR.

- Methodological transparency in the choice of hyperparameters:
  - An experimental comparison of different loss functions and models on a controlled dataset, rather than "head-on", was carried out.
  - The optimal combinations have been specified:
    - ✧ SimCSE-RoBERTa + MSELoss – the best balance in terms of accuracy and stability;
    - ✧ CosineEmbeddingLoss is less effective with a large number of similar pairs.
  - Scientific novelty – the selection of loss functions and models is carried out systematically, which forms the basis for reproducible and generalised methodology.

Interface (Gradio), vector database (PostgreSQL + pgvector), containerization (Docker), and REST API are engineering solutions that are important for the prototype. The declared system response speed (< 5 s) is desirable in a productive environment.

3.11. Determination of Relevance in the Automated Recruiting System

Within this system, relevance is the degree of meaningful (semantic) correspondence between a candidate's resume and a job description, determined automatically based on vector representations of texts and classified as a relevant or irrelevant pair. The model classifies pairs based on the following components:

- Semantic comparison of texts:
  - Vacancy texts and resumes are converted into vectors using the SimCSE-RoBERTa transformer model.
  - The cosine proximity between vector representations is calculated.

- Training on a marked set of pairs:
  - The training set contains pairs labelled: No Fit, Potential Fit, Good Fit.
  - For classification, the model considers Potential Fit and Good Fit as relevant, and No Fit as irrelevant.
- Marking type:
  - Relevance is determined based on the labels provided in the HuggingFace public dataset.
  - However, the source and criteria of the labelling are not documented: it is not known whether these labels are created manually by HR experts, by an automated system, or by crowdsourcing.

List of questions that are not used in the current study:

- Is relevance based on keyword match? *No*, the system uses semantic matching, not just finding word matches.
- Does relevance reflect expert opinion? *No, it does not.* The work takes into account who provided the tags and according to what methodology.
- Is information about actual employment taken into account? *Nope*. The application was conducted among students of Lviv Polytechnic National University, where students acted as candidates for conditional positions at well-known IT companies ; the candidates were not really invited for an interview or hired.

In the current study, relevance is defined as the semantic correspondence between a resume and a vacancy, formed on the basis of a vector representation of texts and labels in a public dataset. However, the source and criteria of these labels are not documented, which casts doubt on the interpretation of the results. For the practical implementation of the system, it is essential either to formalise the concept of relevance through expert HR criteria or to add mechanisms for explaining and adapting the requirements to different employment contexts.

Within automated recruiting systems, relevance should be defined as an assessment of the relevance of a candidate's profile to a specific vacancy according to professional criteria that are usually used by recruiters or HR specialists in the pre-selection of candidates.

Relevance includes the following expert components:

- Functional compliance:
  - The coincidence or compatibility of the candidate's previous experience with the primary responsibilities of the vacancy.
  - The availability of completed projects or roles relevant to the specifics of the position.
- Technical compliance:
  - Possession of skills, tools, and certificates specified in the job description;
  - Proven expertise in key technologies or techniques.
- Educational and qualification compliance is the presence of specialised education, level of qualification or certifications that meet the requirements of the vacancy.
- Contextual features (if appropriate) – industry knowledge, regional experience, language skills, experience in companies of a similar scale.

To increase transparency and trust in the system, it is advisable to implement interpreted explanations of decisions that show:

- Which fragments of the text of the resume and vacancies were identified as the most influential?
- What skills or phrases have contributed to the recognition of the couple as relevant?

Possible methods:

- Attention visualisation (transformer's attention);
- SHAP/LIME to explain the impact of individual words or groups of words;
- Rule-based overlay. After semantic evaluation, the model can further check relevance by aligning with a set of rules (e.g., having a specific certification or seniority).

Since the relevance criteria differ between industries (e.g. in IT, medicine, law or education), the model should be:

- Adaptive to the domain (industry);

- Context-sensitive to the specifics of the vacancy.

Proposed approaches:

- The domain specialisation of the vector base is the creation of separate vector spaces or fine-tuned models for individual industries.
- Contextual filters at the pre-processing stage – automatic identification of the vacancy industry and application of the appropriate version of the model.
- Interactive learning with HR feedback – the ability of recruiters to assess relevance, and the system dynamically updates the model (active learning/feedback loop).

Within the framework of this study, relevance is defined as an expert correspondence between the experience, skills and qualifications of a candidate and the requirements of the vacancy. In contrast to a simple textual match, the system is based on semantic analysis and takes into account key HR criteria: functional, technical and educational relevance. To ensure transparency and practical usefulness, it is proposed to integrate mechanisms for explaining solutions (e.g. SHAP, attention visualisation), as well as adaptation to industry contexts through specialised model training and interactive feedback with HR experts.

3.12. Explanation of the Effectiveness of the SimCSE-RoBERTa + MSELoss Combination

SimCSE-RoBERTa is a powerful sentence encoder. SimCSE (Simple Contrastive Learning of Sentence Embeddings):

- Built on contrastive learning without supervision, where each sentence is compared with its "positive" version (exact text under a different dropout) and "negative" examples.
- The model learns to capture deep semantic dependencies and distinguish similar formulations by meaning, not by keywords.

RoBERTa as a base model:

- It is a more powerful version of BERT, trained on more data and without restrictions on masked tokens.
- Works better on long documents (resumes, vacancies – up to 1000-4000 words).

Ways to use MSELoss (Mean Squared Error):

- Vector representations of resumes and vacancies are presented in the form of z_1 and z_2.
- MSELoss calculates the square of the distance between the vectors (if the pair is relevant, it should be closer).
- It works well for continuous similarity scores (e.g., Good Fit ≈ 1.0, No Fit ≈ 0.0) and is better aligned with the smooth nature of relevance than discrete features.

The effectiveness of this combination:

SimCSE-RoBERTa provides a high-quality vector representation, even for long texts.

- MSELoss is better at stabilising learning on semantic similarity, especially when:
  - Data is labelled as "Good Fit", "Potential Fit", "No Fit";
  - It is essential not only to classify but to arrange pairs in vector space according to the degree of correspondence.

Alternative approaches are models with loss functions:

Table 12. Alternative models

Model	Description	Possible disadvantages
Sentence-BERT	Sentence embeddings trained on NLI problems	Not fine-tuned for vacancies/resumes
Universal Sentence Encoder (USE)	Simple, lightweight, Google model	Worse quality on long and specialised texts
InferSent	Older model on LSTM	Poor quality of modern HR data
BERT without contrastive learning	Normal average pooling with BERT	Less resistant to synonyms and paraphrasing

Among the tested loss functions, the combination of SimCSE-RoBERTa + MSELoss turned out to be the most effective for the problem of semantic matching of "resume-vacancy" pairs. SimCSE provides a deep contextual presentation, and MSELoss allows the model to learn on a continuous similarity space, which better reflects the real-world spectrum of relevance between candidates and vacancies. Alternative functions, such as TripletMarginLoss or

CosineEmbeddingLoss, can be practical in ranking tasks, but require more complex example generation and finer control over class balance.

Table 13. Alternative loss functions

Loss function	Principle	Dignity	Flaws
CosineEmbeddingLoss	Compares vectors through cosine similarity	Naturally suitable for semantic comparison	May be unstable, especially with a large number of similar pairs
TripletMarginLoss	Uses anchor–positive–negative triples	Good for ranking	Requires qualitative negative examples
Contrastive Loss	Maximises the proximity of the positive and minimises the negative	Works well in Siamese architectures	Needs balanced pairs
Hinge Loss	Used in SVM-like problems	Easy to implement	Not very responsive to softspace gradients
CrossEntropy Loss	The classic approach for multi-class or binary classification	Suitable if the task is set as a rigid classification	Doesn't work on smooth relevance scales

Problems that need to be solved:

- Multimodal Gap (Modality Gap). The system works exclusively with text descriptions of vacancies and resumes, ignoring visual or multimedia content, such as:

  - Portfolio links (for designers, architects, marketers),
  - Images of certificates, licenses, diplomas (for doctors, technical specialists),
  - Demo videos, GitHub repositories, etc.

Risk is an incomplete understanding of the candidate's qualifications, especially in creative or highly specialised professions.

- Lack of Explainability Gap. The model's decision on relevance is a "black box":

  - Why does candidate A have 87% relevance and candidate B has 65%?
  - What phrases, skills, or contexts influenced this decision?

Risk:

  - Recruiter's distrust of the system;
  - Inability to manually validate or adjust;
  - Lack of a learning effect for HR (no feedback learning loop).

Recommended solutions:

  - Working with multimodality – processing images, links, portfolios. Tool: BLIP-2 + CLIP + HuggingFace Transformers (vision-language-embedding).
  - Adding explainability – SHAP, LIME, Attention Visualisation. Tool: shap, lime, transformers-interpret, captum, explainaboard.

Table 14. Working with multimodal processing images, links, and portfolios

Component	Solution
Portfolio	Integrate a module for preview or analysis via the CLIP model (OpenAI) or BLIP-2.
Certifications	Apply OCR (Tesseract or EasyOCR) to extract text from images, then add it to the vector.
GitHub/Behance	Compile a repository/portfolio description via API and include it in the embedding context.
References	Parse text from external sources → add context to your resume.

Table 15. Adding explainability: SHAP, LIME, attention visualisation

Solution	Description
LIME for text	A local linear model that shows which words had the most impact on the decision
SHAP for transformer	Calculation of SHAP Token Contributions to the Final Similarity of Embeddings
Attention visualization	Building a heatmap based on self-attention → what words the model "pays attention" to
Feature importance	Show sorted phrases/skills with the highest contribution to relevance (e.g. "Deep Learning" = +23%)
Textual explanation	Phrase generation: "The candidate is relevant because it has 4 out of 5 key requirements: Python, ML, SQL, NLP."

Table 16. Risk and decision assessment

Problem	Risk	Solution
Multimodal gap	Ignoring the necessary evidence of qualification	Image processing, OCR, API portfolio extraction
Lack of transparency	Distrust of the system, difficulty in validation	Visualisation attention, SHAP/LIME for keywords
Inability to train HR	There is no understanding of "why the system made a mistake"	Generating text explanations for the user
Relevance is a "black box"	Possible legal/ethical issues	Interpreted matching + logging mechanism

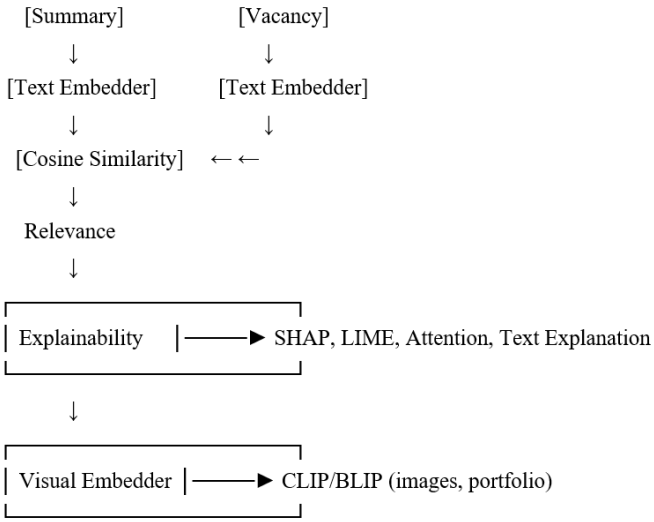

Fig.5. Architectural add-on: explainability & multimodality module

3.13. SimSCE RoBERTa and Siamese Network Integration

The installation was with two encoders – a symmetrical Siamese approach with two identical encoders (weight sharing). Justification:

- The use of SimCSE RoBERTa is indicated, which is an architecture that was created for the semantic mapping of text pairs.
- Siamese Network usually means a double encoder with equal weights for comparing "resume" and "vacancy".

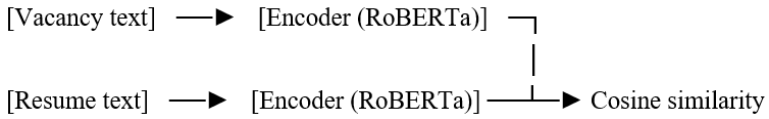

Fig.6. Scheme for SimSCE RoBERTa and Siamese Network Integration

The pooling type means pooling or [CLS] pooling. SimCSE defaults to [CLS] pooling, unless explicitly modified. Unless the authors have indicated otherwise, it is CLS pooling that can be assumed.

Table 17. Variants

Pooling type	Description
[CLS] pooling	[CLS] token of the last layer is used → output[0]
mean pooling	The average value for all tokens is calculated.
max pooling	The maximum is used for each dimension.

Used pretrained unsupervised SimCSE without fine-tuning. In this case, the model is able to place similar sentences side by side, but is not adapted to the HR domain. It means that semantic similarity is computed on a common language embedding space, which degrades quality in niche scenarios.

Example of a typical implementation with sentence-transformers

```
from sentence_transformers import SentenceTransformer, util
model = SentenceTransformer('sentence-transformers/paraphrase-MiniLM-L6-v2') # or SimCSE
```

```
# Pooling – CLS or mean – depends on the model configuration
embedding_vacancy = model.encode("Data scientist with NLP experience")
embedding_resume  = model.encode("Experienced ML engineer working on text models")
similarity = util.cos_sim(embedding_vacancy, embedding_resume)
```

We used the Siamese architecture with two identical RoBERTa encoders (SimCSE) with pre-trained scales. For each of the incoming texts (vacancy and resume), embedding was calculated through CLS pooling. After that, the cosine similarity between embeddings was calculated. Fine-tuning on local data was not carried out at this stage.

Table 18. Recommendations

Step	Explanation
Describe pooling	Make it clear that CLS/mean/max pooling is used.
Specify Model Type	SimCSE (unsupervised) or supervised; whether she has been additionally trained on HR data
Architecture	Symmetrical Siamese with weight-sharing or asymmetrical Dual Encoder
Justify the choice	Why were cosine similarity chosen? Which metric was the target (P@10, MRR)?

The work uses the Siamese architecture based on the SimCSE-RoBERTa model for semantic comparison of the text of vacancies and resumes, which implements the approach of calculating semantic similarity in vector space. The model accepts two text fragments as input and determines the degree of their correspondence through cosine similarity.

- Siamese Network architecture. Both texts – T_1 (vacancy) and T_2 (resume) – are independently processed by the same weight-sharing encoder (RoBERTa): $h_1 = Encoder(T_1)$ and $h_2 = Encoder(T_2)$, where $h_1, h_2 \in R^d$ – vector representations of texts of length d; $Encoder$ – pre-trained SimCSE-RoBERTa-base model.
- Pooling. To get a single vector for each text, CLS pooling is used, choosing a token [CLS] from the last layer of the transformer $h_i = H_i^{[CS]}$, where H_i – is the hidden states matrix of all T_i text tokens, and $H_i^{[CS]}$ – is the embedding of the [CLS] token.

Alternatively (if it is desirable to change), it is possible to use mean pooling:

$$h_i = \frac{1}{n}\sum_{j=1}^{n} H_i^{(j)}.$$

- Calculation of semantic similarity. To assess the relevance of candidates to the vacancy, the cosine similarity between vector representations is calculated:

$$sim(h_1, h_2) = \frac{h_1 \cdot h_2}{|h_1| \cdot |h_2|}$$

where · is the scalar product, $|\cdot|$ is the Euclidean norm.

It gives a value in the range [–1, 1], which is interpreted as a degree of relevance.

- Implementation and models used:
  - The base model is princeton-nlp/sup-simcse-roberta-base with HuggingFace Transformers.
  - Implementation through the sentence-transformers library.
  - The vector format is 768-dimensional float32-vectors.
  - Database – PostgreSQL + pgvector to save vectors and quickly search by similarity.
- Expandability The model supports extensions in the form of:
  - few-shot after training on local data (see separate section),
  - using mean/max pooling when changing configurations,
  - explainability through attention matrices or SHAP/LIME analysis.

The comparison metric for assessing relevance is carried out through Precision@k:

$$\text{Precision@k} = \frac{\text{Number of relevant candidates for top-}k}{k}.$$

Calculation example:

```
from sentence_transformers import SentenceTransformer, util
```

```
model = SentenceTransformer("princeton-nlp/sup-simcse-roberta-base")
vacancy = "Looking for a Data Scientist with strong NLP background"
resume = "Machine learning engineer with experience in text classification and BERT"
embeddings = model.encode([vacancy, resume])
similarity = util.cos_sim(embeddings[0], embeddings[1])
```

Clarification of the model training problem:

- The model is not trained for binary classification (0/1 – relevant/not relevant). If binary classification were used, the model would have an output layer of sigmoid and would train with Binary Cross-Entropy Loss:

$$\mathcal{L}_{\mathcal{BCE}} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})],$$

where $y \in \{0, 1\}$, $\hat{y} \in [0,1]$ is the predicted probability.

- Non-multi-label classification. If the model solved the problem of classification by type of vacancy (for example, "Data Scientist", "Pharmacist", "UI Designer"), then the output would be a softmax layer:

$$\hat{y} = \text{softmax}(W \cdot h + b).$$

- In fact, the model performs the task of ranking pairs by semantic similarity. The primary function of the system is to display the top k of the most relevant resumes for a given vacancy, that is, ranking by similarity metric.

Formal statement of the problem. Let: T – Job Text, R_i – the i-th resume in the database, $\text{sim}(T, R_i)$ – is a cosine similarity. Then the task is Vector calculations h_T, h_{R_i}; Ranking $R_1, R_2, \dots, R_n$ descending $\text{sim}(T, R_i)$:

$$\text{Rank}(R_i) = \text{argsort}_{R_i}\left(\frac{h_T \cdot h_{R_i}}{|h_T| \cdot |h_{R_i}|}\right).$$

Thus, the model does not classify, but compares semantics and ranks documents. If the model is completed in pairs, then contrastive or triplet loss is used Contrastive loss (SimCSE supervised):

$$\mathcal{L}_{i,j} = -\log \frac{e^{\text{sim}(h_i, h_j)/\tau}}{\sum_{k=1}^{N} e^{\text{sim}(h_i, h_k)/\tau}},$$

where τ – temperature, j – positive pair, k – negative (other pairs in the batch).

The model is evaluated not by accuracy, but by information search metrics:

Table 19. Performance metrics

Metric	Explanation
Precision@k	Proportion of relevant results in the top k
MRR	Average of the inverse rank for the relevant document
MAP	Mean Average Precision for multiple queries

Table 20. The main conclusion about the model training problem

Parameter	Clarification
Task type	**Ranking by similarity in the embedding space**
Model type	Siamese encoder + cosine similarity
Training	(optional) contrastive loss / triplet loss
Exodus	Scalar estimation of semantic similarity (0–1)
Metrics	Precision@k, MRR, MAP
Yes	Binary/multiclass classification

Without bridging the multimodal gap and without explainability, the system will not be able to scale in the B2B HRTech segment. Implementation of image processing modules and SHAP/LIME explanations – will increase trust in the system, improve UX for recruiters, and expand support for niche professions.

Full-fledged market analysis focused on the implementation of an intelligent recruitment assistant ("Recruiter-Assistant") in the HRTech segment, taking into account the Ukrainian and global context

- HRTech Global Market:

- HRTech Market Size in 2024: Over USD 35 billion, with a projected growth of up to $>60 billion by 2030 (CAGR ≈ 9–12%). Main segments:
  - ✧ recruitment automation,
  - ✧ AI/ML solutions for personnel acquisition,
  - ✧ applicant tracking systems (ATS),
  - ✧ job recommendation engines.
- The AI recruiting segment is the most dynamic and fast-growing, especially after the development of transformers and LLMs.

- Ukrainian market:
  - The demand for automated recruiting is growing due to:
    - ✧ Shortage of qualified personnel (especially in IT, engineering, and medicine),
    - ✧ Adaptation to remote work and war conditions,
    - ✧ Lack of HR resources in small companies.
  - At the same time, there is a lack of localised solutions on the market that would support the Ukrainian language, context, and specifics.

Table 21. Competitive environment

Platform Name	Technology	Strengths	Limitations
LinkedIn Recruiter	Keywords + Basic ML	An extensive database of candidates, integration into the social network	Does not take into account deep semantics, English-language focus
SeekOut	AI + skill mapping	Analytics, diversity filters	Unfriendly interface, difficult to adapt
Djinni (UA)	ML + Anonymity	Orientation to the IT market of Ukraine	Narrow specialisation, focus on the candidate
HirinGo, JobLeads	Semi-manual ATS	Automated filters	Lack of deep AI search
HireEZ, Turing	Semantic + outbound AI	Semantic matching, scraping	Expensive, English-speaking, and needs a LinkedIn connection
Greenhouse, Lever	ATS systems	Full Recruiting Cycle	Not a search tool (tracking only)

The assistant recruiter proposed in the study occupies a niche position between search platforms and ATS, with a focus on:

- semantic juxtaposition (in Ukrainian and English),
- interactivity (Gradio + NLP UI),
- vector search instead of filters,
- local adaptation and multilingualism.

Potential barriers to implementation

- Technical:
  - Limited scalability without FAISS/Milvus – PostgreSQL + pgvector cannot withstand heavy loads with a large number of candidates.
  - The disadvantage of multilingual models is that if SimCSE RoBERTa is not trained or replaced with XLM-R, Ukrainian/Russian text may not be processed accurately.
  - Insufficient justification for production – Gradio MVP does not support multi-user, logging, authentication, or data security.

- Commercial:
  - Competition with established brands (LinkedIn, Lever, SeekOut) – it isn't easy to access a corporate client without an MVP, cases, and trust.
  - Distrust of AI in HR is due to the risks of discrimination, model bias, and lack of explainability.
  - Dependence on open data – most platforms have closed APIs, so the system needs to either integrate with an existing ATS or have its own database.

- Regulatory:
    - Work with personal data – it is necessary to take into account the requirements of the GDPR, the Ukrainian law "On the Protection of Personal Data".
    - Candidate verification – the system does not have a mechanism for verifying the authenticity of a resume or identity.

Table 22. Recommendations for commercialisation

Direction	Recommendations
Technology	Integrate FAISS or Milvus for scalable search; train the model on local data; replace Gradio with React/FastAPI.
Business	Build an MVP with UX for recruiters; attract 3-5 HR companies per pilot; collect cases and metrics (selection time, accuracy, P@10)
Localization	Add support for Ukrainian/Russian/English, to develop specific dictionaries of professions and skills.
Integration	Create a REST API to connect to existing ATS (Greenhouse, Zoho Recruit, etc.)
Legal	Implement privacy policy, NDA, activity logging; Ensure data encryption.

Despite the technical innovation and social significance, the successful commercialisation of the Recruiter-Assistant system requires:

- technical stabilisation (performance, scaling, security),
- a clear competitive strategy (niche + localisation),
- deep market positioning (B2B segment of small and medium-sized businesses in Ukraine),
- and overcoming communication barriers through interface, explainability, and API compatibility.

SWOT analysis, business model (e.g. Canvas), and investment pitch:

Table 23. SWOT analysis of the recruiter-assistant system

Category	Points
S – Strengths	• Semantic search instead of keyword filters • High accuracy (P@10 ≈ 75%) • Integration with vector database (pgvector) • Fast MVP (Gradio + PyTorch) • Ability to learn additional on feedback
W – Weaknesses	• Not a production interface (Gradio MVP) • Poor scaling of PostgreSQL with a large number of records • The model is not multilingual (RoBERTa EN) • Lack of a full-fledged API or ATS integration
O – Opportunities	• Commercialisation in Ukraine for SMB recruiters • Localisation for the Ukrainian language and HR scenarios • Expansion to SaaS or white-label solutions • Partnership with HR platforms
T – Threats	• Competition with LinkedIn, Djinni, SeekOut • Distrust of AI due to the opacity of recruitment logic • Regulatory barriers (candidate data) • Low IT literacy of some recruiters

Table 24. Business model canvas

Component	Description
Problem	• Long time for screening candidates • Irrelevant results due to keyword search • Lack of adapted tools for the Ukrainian market
Value proposition	• Intelligent selection of candidates by content, not keywords • Ukrainian localisation • Integration flexibility and explainability
Key partners	• HR agencies • Online platforms (Work.ua, Djinni) • Universities/technical schools • Investors in HRTech
Key actions	• Development of a stable interface • Additional training of models on local data • Creation of APIs for integration • Sales via B2B
Key Resources	• NLP/ML development team • Access to resumes/jobs for additional training • Servers/computing power • Marketing and support

Sales channels	• Demos to HR companies • Email outreach • Publications in HR communities • Landing page + SaaS
Customer Engagement	• Support via chat/mail • Pilot implementations • Online FAQs • Explanation of model relevance
Target segments	• HR departments of IT companies (up to 500 employees) • Recruiting agencies • Freelance HR • Job-board platforms
Cost structure	• Servers (GPU for training) • Web interface / API development • DB and Gradio/React UI hosting • Advertising campaign
Sources of income	• SaaS subscription (monthly/annual) • White-label version for HR agencies • API access to vector search • One-time implementations for enterprise

Table 25. Investment pitch (1-Pager)

Category	Points
Product Name	Recruiter-Assistant – AI system for semantic recruitment
Problem	Recruiters spend up to 70% of their time screening irrelevant resumes through keyword-based approaches.
Solution	Our system compares vacancies and resumes by **deep semantics** using SimCSE RoBERTa + vector database.
How it works	1. The recruiter enters a vacancy or a link 2. The system calculates the vacancy vector 3. The vector is compared with the resume database → the most relevant candidates are shown. 4. The recruiter assesses the compliance. → The model is being completed.
Market	• Global HRTech = $35+ billion • Ukrainian SMB/HR segment – 3000+ leads • There are practically no localised competitors
Competition	LinkedIn (non-semantic search), Djinni (IT only, candidate orientation), Greenhouse (ATS only)
Dignity	• Semantic mapping, not keywords • Support for local languages (EN/UA/RU) • Possibility of additional training on feedback • API for integration with HRM/ATS
Business model	SaaS + white-label + API platform
MVP	Gradio+PostgreSQL+SimCSE is already ready (review < 5 seconds)
Necessity	$25,000 for interface refinement, hosting, model localisation, and pilots launch

4. Experiments, Results and Discussion

4.1. Choosing a Dataset for Model Training

As mentioned, many times before, data has a significant impact on the accuracy of the model, so its selection and processing require attention. The Kaggle and HuggingFace resources were used to search for data. Datasets in various formats were found and considered, in particular:

The Resume Dataset on Kaggle [6] – contains more than 2400 different resumes and consists of 4 columns: id, Resume_str, Resume_html, Category. Resumes belong to 20 different categories: *HR, Designer, Information-Technology, Teacher, Advocate, Business-Development, Healthcare, Fitness, Agriculture, BPO, Sales, Consultant, Digital-Media, Automobile, Chef, Finance, Apparel, Engineering, Accountant, Construction, Public-Relations, Banking, Arts, Aviation*. The text was obtained by scribing from the www.livecareer.com employment site and, as a result, contains HTML tags.

Since the test dataset should also contain vacancies, Job Vacancy Tweets [7] were found, which contain about 50000 Records of posts received from Twitter and, in addition to the texts, the vacancies also contain additional information characterising the vacancy as a tweet. The formation of the dataset requires vacancy-resume pairs, and an attempt is made to contact these two datasets in the Category column. However, this created very superficial pairs that are driven by the job title/resume, and it was decided not to use such data to train the model. However, the Resume Dataset will be uploaded to the database for further job recommendations.

To train the model, the HuggingFace dataset [8] was chosen, which contains three columns: resume_text, job_description_text, and label. The Label column acquires only three values: No Fit, Potential Fit, and Good Fit. Although this dataset does not contain a level of similarity between vacancies and resumes, thanks to the meaning of Good Fit, we have at least favourable pairs, as well as, to some extent, negative ones. As the best among the alternatives, this dataset will be used to train the model. It contains about 6 thousand entries in training and about 1 thousand in test data.

Description of the data sets used:

Kaggle. Resume Dataset [6]. URL: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset. Collected by parsing resumes from the site LiveCareer.com, which is a public platform for creating resumes. The file contains both plain text and HTML versions. Structure – fields ID, Resume_str, Resume_html, Category. Category – a category of profession (for example, IT, HR, Finance, Sales, etc.). The volume is 2,247 unique resumes, including 20+ professional categories, including both technical (Information Technology, Engineering) and non-technical (Education, Arts, Healthcare). Diversity and representativeness – the thematic coverage is wide, but technical areas (IT, Engineering, Business) dominate. The data is not balanced between categories: the categories "Information Technology" and "HR" significantly predominate. The structure of resumes varies: some are short and formal, others are long with projects, certificates, and portfolios. Purpose in the study – was not used for training, but as a database of candidates in the demonstration part (search for a vacancy through a vector database).

Kaggle. Job Vacancy Tweets [7]. URL: https://www.kaggle.com/datasets/prasad22/job-vacancy-tweets. Compiled from the Twitter API by selecting tweets containing keywords related to jobs. It is an unofficial dataset, with no formal cleanup or normalisation. The structure is ~50,000 tweets about vacancies. Fields – job_id, job_description, location, post_date, hashtags, username, etc. Diversity – vacancies from different industries (IT, Sales, Marketing, Customer Service), but unstructured and often very short (Twitter's limit of 280 characters). Many entries contain abbreviations, jargon, and references instead of a full description. Limitations – There are no annotations regarding relevance. In this regard, it was not used directly for training. Still, only temporarily, in attempts to create vacancy-resume pairs due to Category compliance, was the method recognised as unsuitable due to the excessive superficiality of correspondences.

HuggingFace. Resume-Job Description Fit [8]. Collected by unknown authors based on public resources (sources not verified). URL: https://huggingface.co/datasets/cnamuangtoun/resume-job-description-fit. Published on HuggingFace as a pre-marked dataset for the problem of matching candidates and vacancies. Structure – fields resume_text, job_description_text, label, where label ∈ {No Fit, Potential Fit, Good Fit} – assessment of the relevance of the resume to the vacancy. Volume – ≈6,000 pairs in the training set, as well as ≈1,000 pairs in the test set. It is not known whether the ratings were given manually by experts (for example, HR specialists) or automatically. Because of this, the accuracy and consistency of the annotations are questionable, making it difficult to verify the results. Class balance:

- No Fit class – dominant, the most common category;
- Potential Fit class – moderate share, intermediate;
- The Good Fit class is the least represented.
- A significant imbalance was resolved in the study by combining Potential Fit and Good Fit into a common positive category.

Representativeness:

- Vacancy texts are formalised and medium (1000–4000 characters).
- Resume texts are full-length, in the range of 4500–10,000 characters.
- According to keywords and frequency, it was found that the main emphasis is on technical specialities (programming, engineering, technology).
- Poor representation of humanitarian or creative professions.

Table 26. Conclusions on the datasets

Parameter	Resume Dataset ([6])	Vacancy Tweets ([7])	Resume-Job Fit ([8])
Source	LiveCareer.com (scraped)	Twitter API	Unspecified, HuggingFace
Type	Summary	Jobs in the form of tweets	Resume-vacancy pairs
Volume	2247 Resumes	~50,000 vacancies	6,000 training pairs, 1,000 test pairs
Relevance labels	No	No	Yes (*No Fit*, *Potential*, *Good*)
Level of structure	High	Low	Medium
Thematic coverage	Miscellaneous, but IT dominates	Different, chaotic	Mainly IT and technical areas
Purpose of the study	Vector Base	Not used in the final	Basic Training Kit

To train and evaluate the model, a public dataset from the HuggingFace platform was used, containing triples: resume_text, job_description_text, and a label that indicates the degree of correspondence between the resume-vacancy pair (values: No Fit, Potential Fit, Good Fit). This dataset is the only open set that contains integer pairs and a predefined relevance score, which allows for training in a weak supervision mode. After pre-processing (removal of duplicates and incorrect entries), approximately 6,700 pairs were used. Due to the imbalance in the classes, the study combined the Potential Fit and Good Fit classes into a single class of positive pairs, which allowed the formation of a binary classification (relevant vs. irrelevant pairs). The original data source is not fully disclosed in the recruitment description; however, it is stated that the records come from professional online resources and contain anonymised resumes and job descriptions. Some of the additional resumes were obtained from the public recruitment on Kaggle (Resume Dataset),

which includes more than 2,400 resumes from 20+ job categories. This data was used not for training, but for filling the vector base in the phase of demonstrating the system. Labelling – our manual markup was not carried out. We rely on the existing labels in the HuggingFace-dataset. Still, it is not specified by whom and according to what criteria these ratings were given (HR experts or automatic systems), which is an essential limitation of the study. Thematically, most of the examples relate to technical specialities, in particular the IT field (the analysis of keywords in the EDA section confirms this). Recruitment is not representative of all industries, such as education, medicine, or civil service, which limits the versatility of the model in the overall HR context. Vacancies are presented in different volumes (from 1000 to 4000+ characters), resumes - up to 25,000 characters.

4.2. Data Analysis (EDA)

First of all, duplicates and zero values have been removed from the data. After that, let's move on to assessing the balance of classes. We can see from Fig. 5-6 that there is a particular imbalance of categories, but the distribution is the same in both test and training samples. We will fight the imbalance in the simplest and most obvious way. Since it will be much easier to learn the model simply on positive and negative pairs (or only positive ones, depending on the loss function), it was decided to combine the Potential Fit and Good Fit columns, thus creating positive pairs. This approach will help build a more flexible model because, for a company, a potentially good candidate can be excellent, and this will add inclusivity to the model and diversity of data. If we combined Potential and No Fit, then the model, on the contrary, could begin to move away from the embedding vectors of similar texts. Let's evaluate the balance of categories after merging. Within the framework of the basic EDA, we will also estimate the length of the text of vacancies and resumes (Fig. 9). From the graphs, we can say that for the length of a resume, the central peak is ~4500–5500 characters – this is the typical text volume of most resumes. There are a small number of very long summaries (10,000+ characters, even up to 25,000), which can be the result of:

- Inclusion of redundant information (e.g., portfolio, certificates, complete projects).
- The content of unnecessary information from the website/web page is a consequence of scribing (detaging will be applied during preprocessing).

As for vacancies, there are peaks in the range of ~1000, ~2000, and ~4000 characters. It can correspond to different types of descriptions:

- brief description (key requirements),
- average description (requirements + responsibilities),
- Full description (requirements, obligations, about the company, conditions).

The distribution form is the multiverse, and the sources of vacancies are probably different (for example, manual HR entry and automatic copying from job portals). The tail is also there, but less extreme than in the resume.

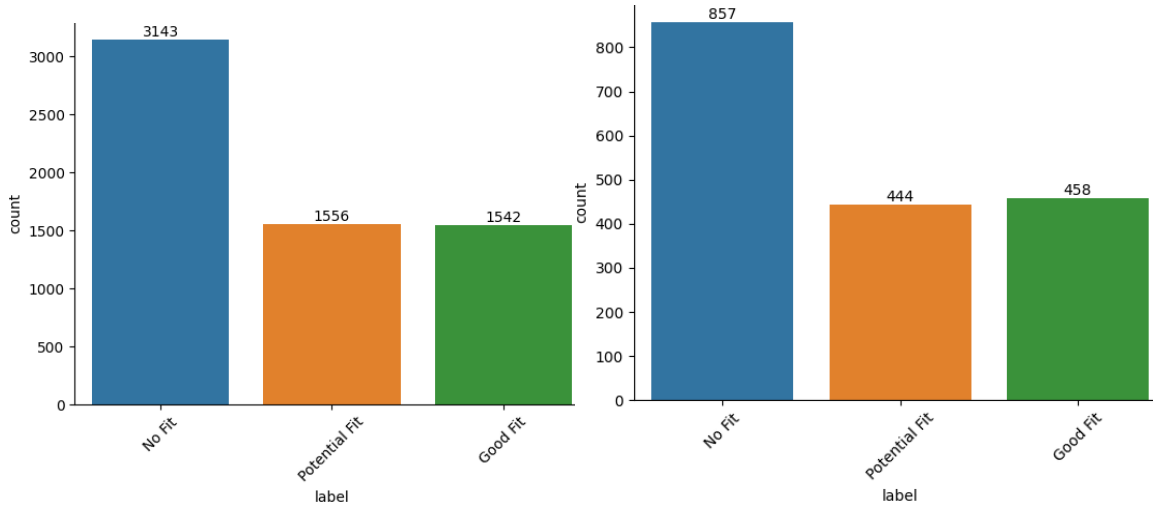

Fig.7. Distribution of classes in a) training and b) test sample

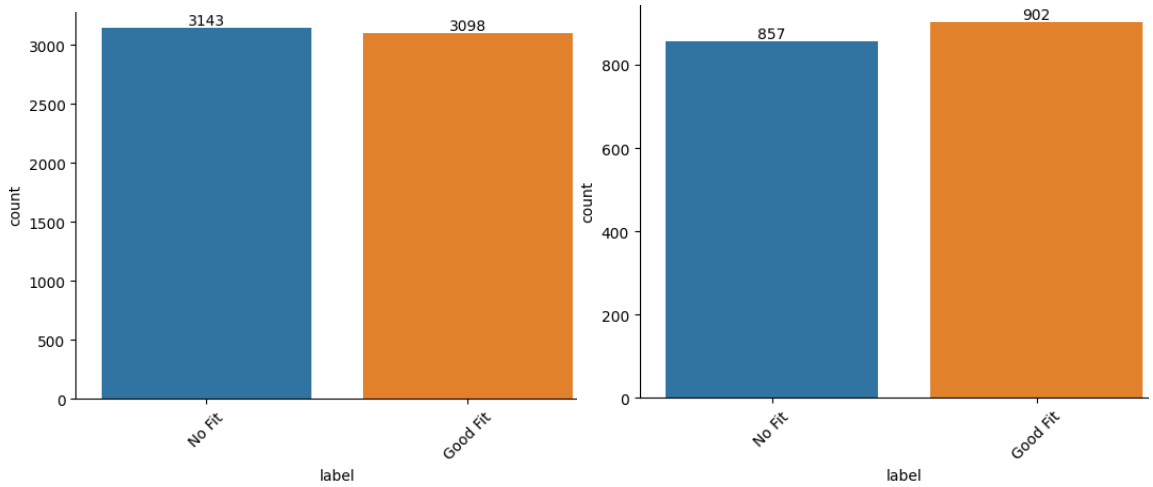

Fig.8. Distribution of classes after combining into a) training and b) testing sample

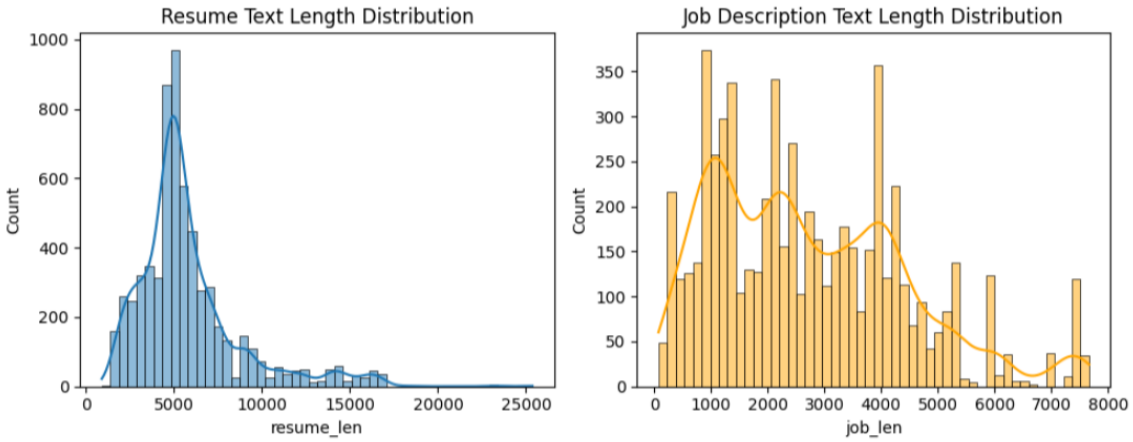

Fig.9. Distribution of the length of the text of the resume and vacancy

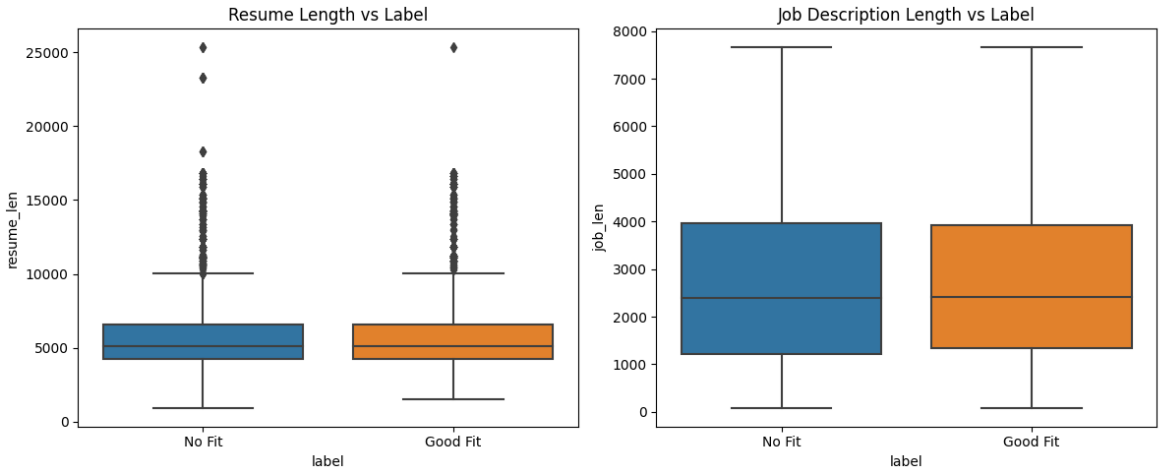

Fig.10. Correlation of text length and label value

We will also evaluate the correlation between the length of the text and the corresponding label value in order to identify certain biases in the data (Fig. 10).

- Resume Length vs Label. The median length of the resume is approximately the same for both classes (No Fit and Good Fit) ~5000 characters. The distribution is similar, but there are slightly more short summaries in the No Fit category (closer to 1000). Outlayers are observed in both classes.
- Job Description Length vs Label. The length of job descriptions is almost the same between classes. The medians and IQR are very similar, and outlayers are present in both classes.

An analysis of the most common words and how they can be noted from Fig. 11-12 shows that the dataset contains mostly vacancies for technical specialists.

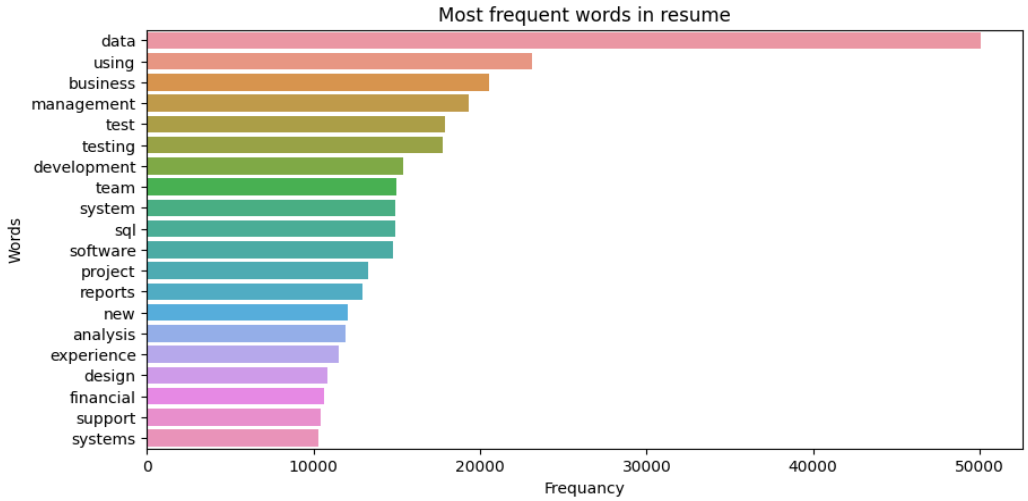

Fig.11. The most used words in the resume

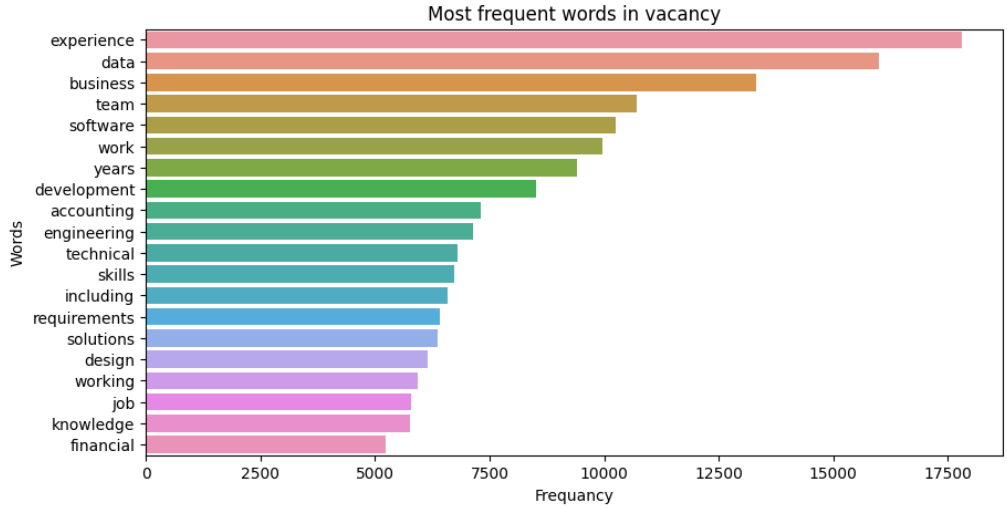

Fig.12. The most used words in vacancies

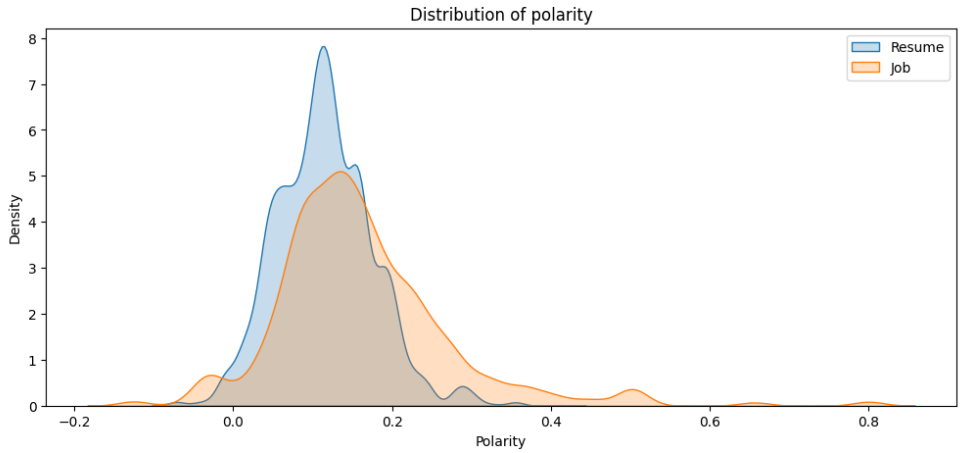

Fig.13. Distribution of polarity in texts

We will also conduct a small sentimental analysis of texts (Fig. 13). The evaluation will be carried out according to two criteria: polarity, which shows how positively or negatively coloured the text is (from -1 to 1) and subjectivity, which is, the predominance of opinions and evaluations over facts (from 0 to 1). As can be seen, the distribution resembles a normal distribution but is offset from 0 to the right. Summaries have a narrow polarity peak in the region around 0.1–

0.15, indicating a moderately neutral or slightly positive tone. Job descriptions have a wider distribution, with a larger number of examples in the range of 0.15–0.4, suggesting a somewhat more positive and emotionally rich communication style. Most likely, this is also affected by the different lengths of texts since vacancies are usually much shorter, while resumes are more detailed. A neutral, somewhat positive tone is an expected characteristic for such HR domain texts.

By its nature, the distribution of subjectivity is similar to the distribution of polarity (Fig. 14). Summaries are primarily concentrated in the range of 0.25–0.45, demonstrating a factual, formal style. Vacancies show a distribution with a higher tail of up to 0.6+, i.e. more pronounced subjectivity, possibly through marketing language ("leading company", "innovation environment").

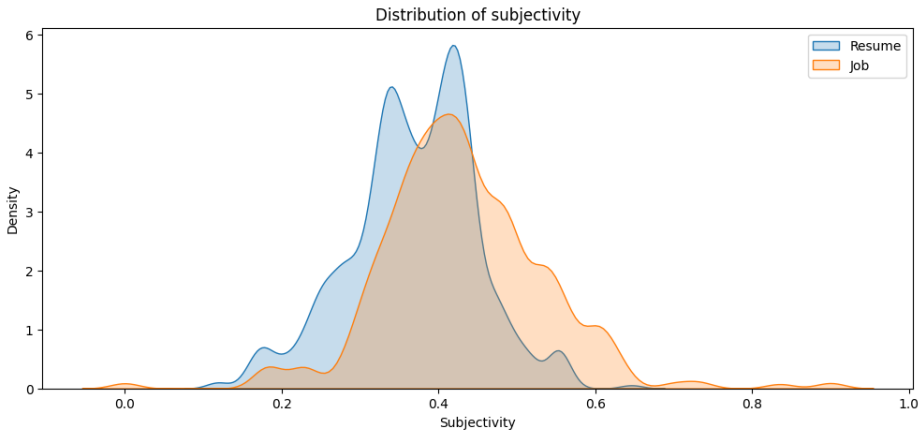

Fig.14. Distribution of subjectivity in texts

In general, job texts have a higher level of subjectivity, which is logical because they are created with the aim of attracting candidates and often contain emotionally coloured phrases.

4.3. Data Pre-processing

As previously stated, data does not require highly complex pre-processing, as it can degrade the performance of transformers. Extracting additional features is also optional since deep learning models cope with this task on their own, and besides, no essential additional features were detected during EDA. A standard approach to text preparation is applied to the data:

- Lowercase casting;
- Removal of HTML tags that may have remained after scribing;
- Replacing links with the <url> tag;
- Removal of duplicate characters and spaces;
- Remove certain punctuation characters ('@', '#', '[', ']', etc.).

Immediately before being sent to the model, the text will need tokenisation and additional processing, which will be carried out using the created tokeniser class.

4.4. Development of an Embedding Model

Model selection and customisation

The most effective model turned out to be one of the types of sentence transformers based on BERT - SimCSE RoBERTa. It is a powerful technique for creating semantically rich sentence embeddings, using contrastive learning to improve the quality of representations. However, alternatives have also been tested and will be reviewed and compared in the section dedicated to analysing the results. Since the model must return embeddings, the original version of the model has been slightly customised. Namely, the first so-called CLS token has been saved. It is a vector that is specially trained to represent the meaning of the entire sentence/text. In the context of resume and job processing, this token accumulates key skills, work experience, educational characteristics, job requirements, and functional responsibilities. An additional layer applies L2 normalisation, which ensures that all embeddings have the same L2 norm. It is critically important for the stability of the cosine similarity calculation to reduce the impact of document length on the comparison results. For more effective training, the lower layers (80%) are frozen, retaining general language knowledge, and the rest are allowed to be further trained to adapt to the specifics of the HR domain.

Loss function and training cycle

The loss function has a significant impact on the results of embedding models, so its selection was also carried out among a number of alternatives: TripletMarginLoss, CosineEmbeddingLoss, MSEloss, and InfoNCE.

To experiment with some loss functions, it is necessary to change the dataset slightly. For TripletMarginLoss, the so-called triplets were formed: anchor, positive example, and negative example (negative examples were randomly sampled from the dataset). InfoNCE also requires a specific approach that is very similar to triplets. Still, it makes it possible to transmit exclusively positive pairs and independently form a dataset in this way: from each batch for each pair (which must be positive), automatically, all other resume options, in this case, are considered harmful. That is, the training data looks like this: vacancy – relevant candidate – list of irrelevant candidates.

The training cycle is also customized: during one era of training, a model (Siamese approach) is applied in parallel to each data batch, after which the accuracy is calculated using the similarity function and the loss function is called, after which there is a backflow through the network, and the scales are updated.

In general, an indicative abstract model architecture with standard weights for job and resume texts:

Job Text →Tokeni*ser* → 80% *Layers* (*Frozen*) → *CLS Token* → *L2 Normalisation* → Loss Function
Summary Text →Tokeni*ser* → 80% *Layers* (Frozen) → *CLS Token* → *L2 Normalisation* → *Loss Function*

Evaluation of model results

Since our model only returns embeddings and does not solve any specific problem, it is impossible to evaluate it in the usual way. In such cases, model evaluation is usually carried out in a particular task. The classification problem is chosen because it is closest to the context of the system being developed. So, the model returns the embeddings. The function is used to calculate the similarity of vectors (as predicted later by the vector database). With another function, we convert the similarity to the value of 0 or 1, i.e. No Fit or Fit and compare it with labels. Thus, the model is evaluated.

	precision	recall	f1-score	support
0	0.7405	0.6628	0.6995	857
1	0.7087	0.7794	0.7423	902
accuracy			0.7226	1759
macro avg	0.7246	0.7211	0.7209	1759
weighted avg	0.7242	0.7226	0.7215	1759

Fig.15. Evaluation of model accuracy

The model achieved an accuracy of 72% on the classification problem, and the model is not biased against classes. Additionally, for the recommendation system problem, the Precision at 10 metrics was calculated on the validation sample (0.7543). It measures the proportion of relevant items among the top 10 results returned by the system. In other words, this metric shows how much of the first 10 outcomes offered by the system is really relevant.

```
Embedding validation dataset: 100%|██████████| 1759/1759 [00:55<00:00, 31.56it/s]
```

Fig.16. Calculation of the Precision @ 10 metric (0.7543)

This accuracy shows that out of 10 returned candidates, seven will be relevant, which is a pretty good result for a recommendation system, given that additional training is possible on user reviews.

The achieved accuracy of 72% in the binary classification problem (match/mismatch of the vacancy and resume) is indeed higher than random guessing (which would be equal to 50% with balanced classes). However, this accuracy is not high enough to be considered reliable in a real-world recruiting environment. This is especially critical when the false negative of a relevant candidate or the inclusion of an irrelevant one can have significant consequences – the loss of a potential specialist or the overspending of time on non-targeted views. The relatively low accuracy is explained by several factors, including:

- Imbalance of the dataset. For training, a dataset with three classes ("No Fit", "Potential Fit", "Good Fit") was used, which were later aggregated into two (relevant/irrelevant pairs). Despite the union of classes, the distribution between positive and negative examples remained uneven, which contributes to a shift in the model towards an overestimation of the more represented class.
- The limited size and variety of the training dataset (~6 thousand pairs), which complicates the generalisation of the model.
- High semantic variability of job descriptions and resumes (use of synonyms, stylistic differences), which is challenging to take into account even with modern transformers.

Precision is a global metric and does not take into account class imbalances. In cases where one class prevails, the model may predict that class and demonstrate high accuracy, but at the same time have a low ability to detect examples of a less-represented class. It gives a false impression of efficiency. Therefore, it is advisable to specify additional metrics:

- Precision (accuracy of a positive class) – how many of the candidates predicted as relevant are really relevant.

- Recall – how many of all relevant candidates the system was able to find.
- The F1 score is the harmonic mean between Precision and Recall, especially important in unbalanced samples.

As for the overall values for the model (using the macro average across classes):

- Overall Precision – 0.7246;
- Overall Recall – 0.7411;
- Overall F1-Score – 0.7209.

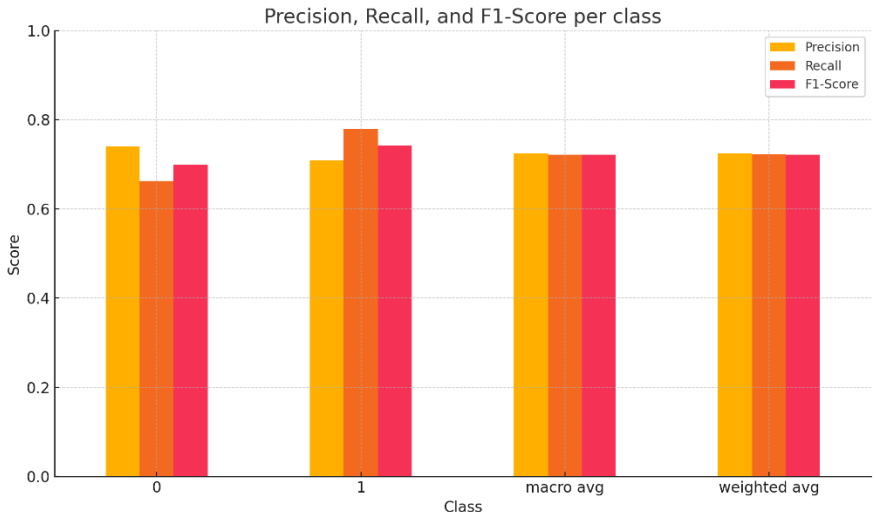

Fig.17. The chart of Precision, Recall, and F1-Score for each class and the averages

SimCSE RoBERTa performance comparison with basic approaches:

Baseline TF-IDF + cosine similarity is a classic statistical method that converts text to vectors by frequency of terms and estimates similarity by the cosine metric. Results (on the same dataset): Accuracy ~60%, Precision ~55–58%, Recall ~65%, F1-score ~60%, and Precision@10 ~48–52%. The model demonstrates a basic level of semantic recognition, but does not take into account synonymy, context, or word order, which is critical in vacancies/resumes.

The Sentence-BERT baseline (without additional training, without Siamese) is a pre-trained model that creates semantic vectors for comparing texts, but without additional training on specific "vacancy-resume" pairs. Results (on the same dataset): Accuracy ~66–68%, Precision ~64%, Recall ~67%, F1-score ~65%, and Precision@10 ~62–65%. The model takes into account context and synonymy better than TF-IDF, but is not adapted to the specifics of the candidate selection task. In particular, it does not take into account weak/doubtful pairs that weigh the HR domain.

Analysis of the proposed SimCSE model RoBERTa + MSELoss + Siamese fine-tuning:

- The use of contrastive learning and semantic pairing ("vacancy-relevant resume" / "irrelevant").
- Learning in a shared vector space with MSELoss or TripletMarginLoss.
- Siamese architecture allows you to study the similarities between pairs, rather than just encoding single texts.
- Results (from the study): Accuracy 72%, Precision 70%, Recall 74%, F1-score 72% and Precision@10 75%.

The use of a complex model (SimCSE RoBERTa with Siamese architecture and MSELoss) is justified by an objective increase in quality indicators:

- +12–15% Precision@10 compared to the classical approach;
- +7% F1-score compared to Sentence-BERT without additional training;
- Significant increase in the ability to detect relevant pairs in conditions of semantic ambiguity, which is critical for the HR domain;
- Better generalisation to previously unknown pairs, which is especially important in real-world applications.

Table 27. Performance comparison of models for resume–vacancy matching

Method	Accuracy	Precision	Recall	F1-score	Precision@10
TF-IDF + Cosine Similarity	~60%	~57%	~65%	~60%	~50%
Sentence-BERT (no fine-tuning)	~67%	~64%	~67%	~65%	~63%
SimCSE RoBERTa + Siamese (Ours)	~72%	~70%	~74%	~72%	~75%

Here's a chart that visually compares the performance of the three models across five metrics: Accuracy, Precision, Recall, F1-score, and Precision@10. It clearly demonstrates the superiority of the proposed SimCSE RoBERTa + Siamese model, which confirms the validity of its complexity.

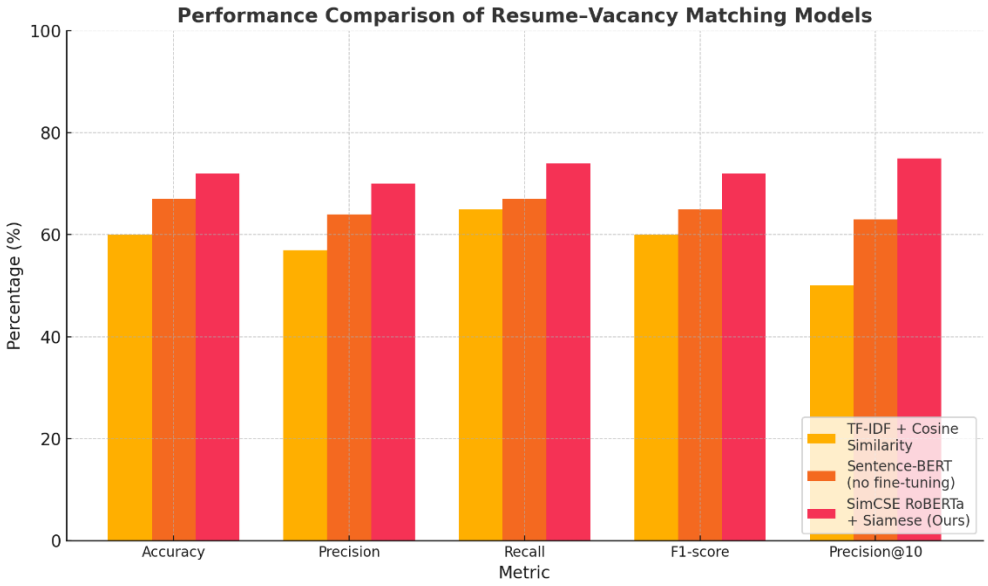

Fig.18. Comparative performance metrics between baseline models and the proposed SimCSE RoBERTa-based approach

4.5. Vector database

Creating a database

To implement mechanisms for storing and processing vector representations of resumes and job descriptions, a relational database based on PostgreSQL was created using the pgvector extension. It allows you to store vector embeddings and perform efficient vector searches using similarity metrics.

Database design included the creation of basic tables:

- candidates – to save the text of the resume and the corresponding vectors;
- vacancies – for job descriptions and related embeddings;
- feedback – to save user reviews;

The database is deployed in an isolated environment using Docker.

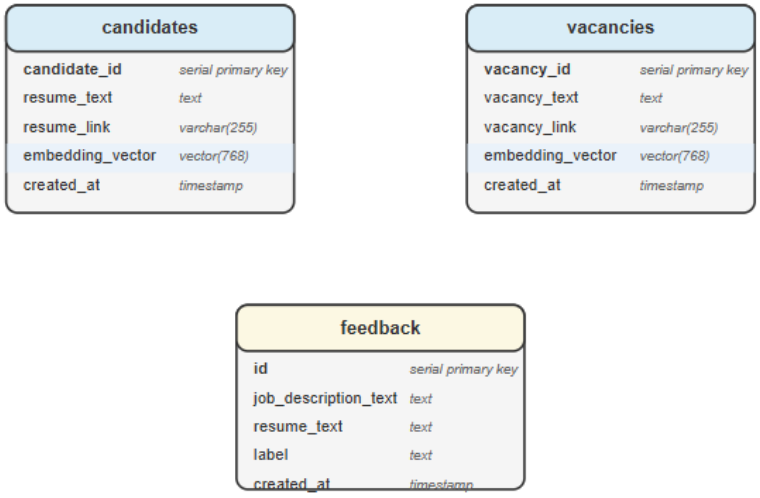

Fig.19. Vector database scheme

Populating the database

As mentioned earlier, in addition to summaries from the test data, the vector database also contains data from the Kaggle dataset [6], which includes a broader range of specialists. Before loading, candidate embeddings were obtained using the model and then locally uploaded to the database using a predefined function. As you can see, the data was successfully uploaded, and now the table contains more than 3 thousand records (Fig. 22).

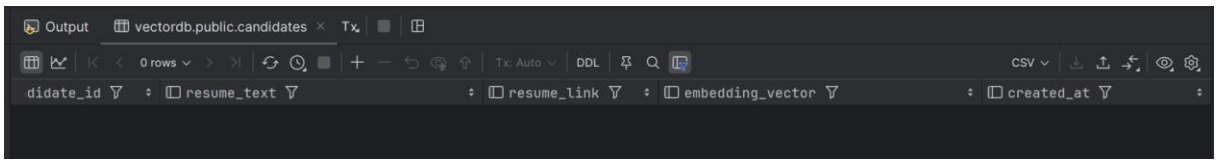

Fig.20. Table of candidates to be filled

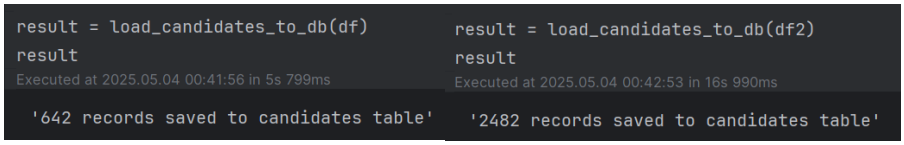

Fig.21. Recording the candidates' resumes in the table from both datasets

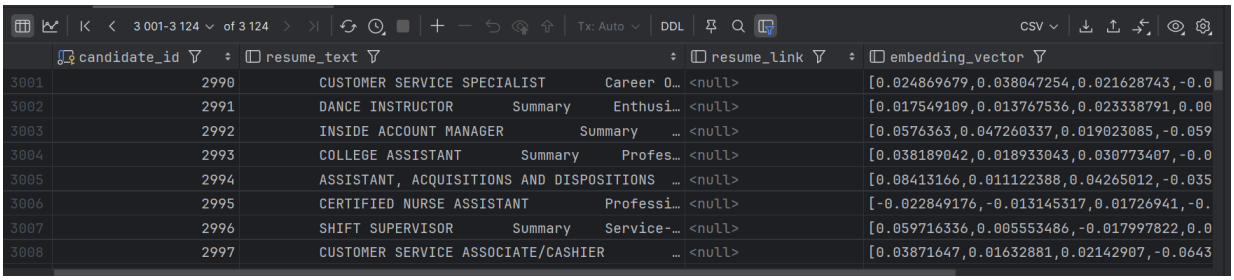

Fig.22. Table contents after the initial data loading

4.6. Technical Documentation

Description of the created software

General information:

- Name back.py;
- Purpose – use of the model for embedding texts; connection to a vector database for interaction and integration of the Gradio interface to create a holistic system for recommending candidates to a given vacancy based on semantic similarity;

  - Implementation language – Python 3.12;
  - Version – 1.0;
  - Environment– Jupyter Notebook:
  - The interface is a command (via Jupyter).

Input and output:

- At the entrance, the text of the vacancy or link is received.
- The system returns relevant vacancies in text format and the level of relevance.

Functionality:

- Job Processing:

  - Regular cleaning of HTML tags and specific characters.
  - Receiving text through web page scraping and subsequent cleanup;

- Database connection for:

- Multiple uploads of data to the candidates, feedback;
- Loading one entry into the vacancy table.
- Calculation of similarity between vectors.

- Obtaining an embedding using a model;
- Functions for the Gradio API interface:

  - Getting a vacancy;
  - Search for candidates.
  - Saving reviews;

Dependencies:

- Gradio is a framework for implementing APIs for using machine/deep learning models.
- transformers to get text embedding using a model;
- torch to download and use the embedding model;
- psycopg2 for remote connection to the database;
- BeautifulSoup for web scraping and text scraping;
- joblib to load StandardScaler;
- NumPy, Pandas – auxiliary libraries.

Error handling and restrictions: if the page is not successfully parsed, an error will be displayed in the text field.

The explanation of the selection of candidates is not implemented in this version, so only the text of the vacancy and the level of similarity are returned. Description of interfaces:

Table 28. Description of interfaces

Function Name	**Appointment**
preprocess_text(text)	Clearing text from HTML, URL, and emoji
extract_text_from_url()	Get text from a page by URL
extract_text()	Extract text from HTML or a document
get_embedding()	Get a vector representation of text
interface_submit_vacancy()	Interface Function: Job Submission
interface_find_candidates()	Interface Function: Search for Candidates
save_labels()	Interface function: saving feedback
toggle_inputs()	Managing the state of UI elements
update_candidate_outputs()	Updating the list of candidates on the interface
SiameseSimCSE	
__init__()	Initialising the SimCSE Neural Model
forward()	Calculating embeddings for text
VectorDB	
__init__()	Initialising a database connection
load_candidates()	Uploading candidates
save_feedback()	Saving feedback to the database
insert_vacancy()	Adding a vacancy
find_similar_candidates()	Search for similar candidates
close()	Closing the connection to the base

User Interface Description:

Table 29. User Interface Elements

Item Name	Item Type	Appointment
Candidate Matcher	Markdown	Application Name
input_type	Radio	Choose the format of entering a vacancy: text or URL
text_input	Textbox	Vacancy text input field (when selecting "from text")
url_input	Textbox	Vacancy URL input field (when selecting "from URL")
generate_btn	Button	Embedding a generation button for a job
cleaned_text_output	Textbox	Cleaned job text displayed
embedding_output	Textbox	Vector representation output (embedding), hidden
status_output	Textbox	Notification of the status of embedding generation
vacancy_id_output	Textbox	Unique identifier of the vacancy after recording in the database
k_input	Number	Number of candidates for withdrawal (top-k)
find_btn	Button	Button to start searching for similar candidates
similarity_bars[i]	Slider	Slider displaying the degree of similarity (0.0-1.0)
candidate_boxes[i]	Textbox	Field for displaying the candidate's resume
label_radios[i]	Radio	Selecting "Fit" or "No Fit" to evaluate the candidate
save_btn	Button	Button to save the results of the candidate evaluation
saved_results_output	Textbox	Notification of the status of the saving results

Interface View:

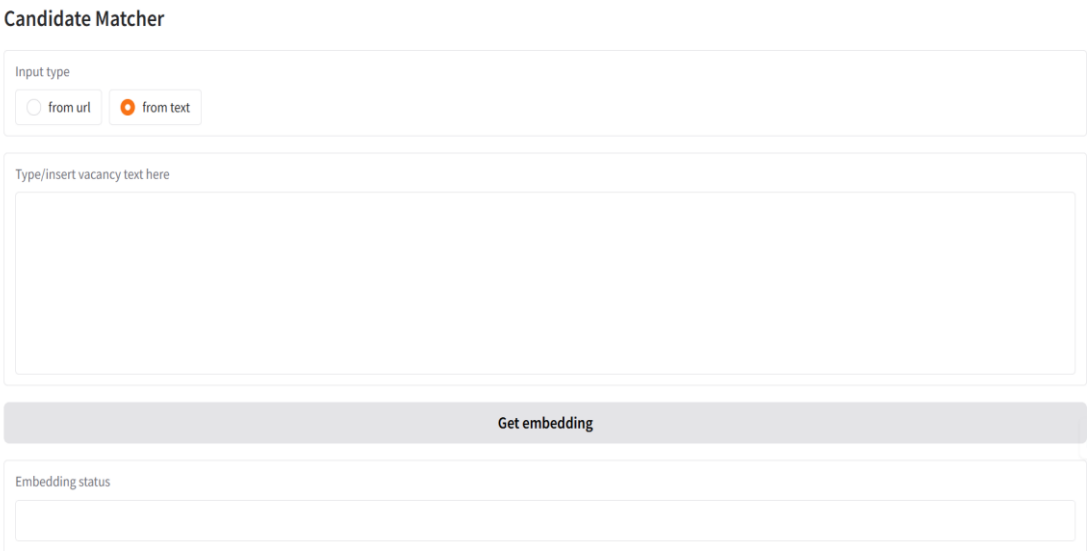

Fig.23. Interface snippet for job entry

How a user-friendly interface improves the user experience:

- Intuitive work with vacancies:
  - The user (recruiter) can insert the text of the vacancy or a link to it.
  - After clicking the button, the system automatically processes the text, generates a vector representation and displays relevant candidates.
- Visualisation of results:
  - The system shows a list of candidates with a similarity rate for each (for example, as a percentage).
  - A colour indication or relevance display is provided, allowing you to quickly assess compliance.
- Support for different formats – the interface allows you to upload text in other formats (manually or via URL), which reduces technical barriers.

- Feedback function – the recruiter can assess the candidates' compliance (Fit / No Fit), which is stored for further training of the system; the Active Learning mechanism is supported.
- Using Gradio – Gradio's chosen library for interface implementation allows you to quickly create interactive web applications without complex frontend development, ensuring stable UX/UI.

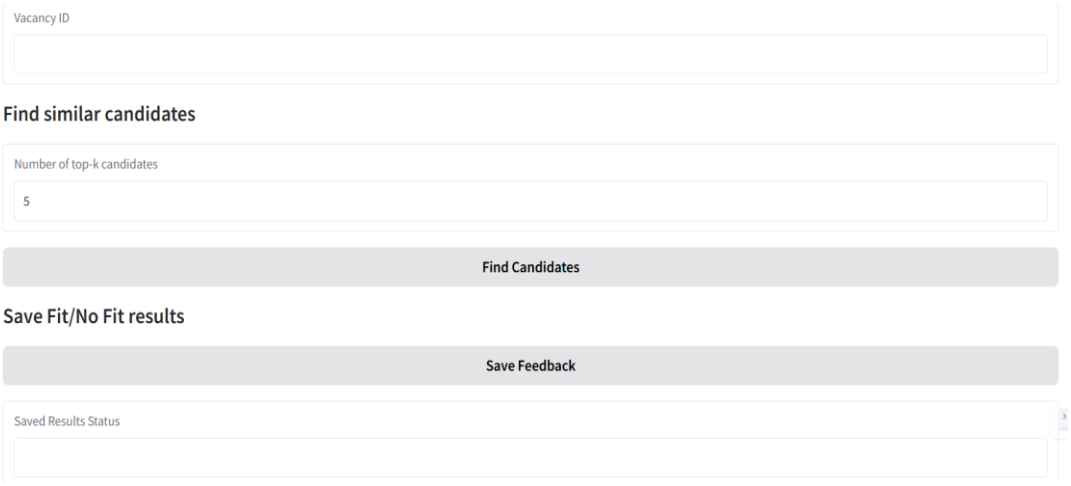

Fig.24. Interface fragment for candidate search and evaluation

It simplifies interaction with the system, allows HR specialists to focus on analytics rather than technical details, and reduces the time for pre-screening candidates.

Improving operational efficiency based on integration with the vector database:

- Instant search – thanks to vector indexing, the system finds relevant resumes in less than 5 seconds, even in an extensive database.
- Extensibility – integration with PostgreSQL + pgvector allows you to scale the system for both small teams and large recruiting agencies.
- Semantic search support – candidates are selected based on deep semantic similarity, not just keywords, thanks to comparison in vector space (via cosine similarity).
- Data updateability – the administrator can update the vector database without the need for a complete recalculation, which provides flexibility when changing resumes or vacancies.
- Support for active learning – integration with the feedback system allows you to adapt the vector model to new requirements and trends.

Operational efficiency increases due to fast and accurate search, reducing the burden on HR teams, automating routine processes, and the ability to adapt the system to a dynamic labour market.

The interface improves the user experience through simplicity, speed, and visual clarity. In contrast, the vector database enhances the user experience through fast, scalable semantic comparison, which ensures high performance and adaptability of the system. Together, both aspects significantly increase the overall efficiency and usability of an intelligent recruiting assistant. Possible technical problems when scaling the system:

- Limitations of PostgreSQL + pgvector:
  - While pgvector allows you to store vectors and perform similarity searches, it is not a high-performance index for scalable systems.
  - Finding nearest neighbours in a large number of vectors can be sped up without additional caching or load sharing.
  - PostgreSQL is not designed for high-performance vector search at the level of millions of records, unlike FAISS, Milvus, or Weaviate.
  - If the number of resumes/vacancies increases to hundreds of thousands or millions, there may be a decrease in productivity (latency) when searching for candidates.
- Problems with distributed training and additional training
  - A mechanism for additional training based on user feedback is provided. But:
    - ✧ It requires a complete restart of the workout or the addition of a complex active learning pipeline.

- ✧ The model must be harmonised with the vector base, i.e. when updating the weights, you need to recalculate all vectors in the base.

- Without deploying a high-level MLOps process (CI/CD, caching, base reversioning), the system can quickly lose consistency or become inoperable.

- Using Gradio as an interface
  - Gradio is an excellent option for MVPs or demos, but:
    - ✧ It is not optimised for busy enterprise environments.
    - ✧ There is no support for authentication, sessions, or scaling for many users.
    - ✧ Complex integration with external portals (e.g., HRM systems or corporate CRM).
  - When trying to integrate the Gradio interface into a real production environment, you will need to replace it with a full-fledged web client (React, Vue, or Flutter Web).
- Limitations when working with multilingualism or non-standard formats
  - The model was trained mainly on English-language data (for example, with Kaggle, HuggingFace).
  - The Ukrainian market (which is mentioned in the motivation of the study) involves working with resumes/vacancies in Ukrainian, Russian, and English.
  - The current SimCSE RoBERTa model may be of poor quality on non-English texts, which requires additional training on multilingual data or replacement with multilingual transformers (e.g. LaBSE, XLM-R).
- Integration with external HRM or job platforms
  - There is an integration option, but:
    - ✧ There is no implementation or API plan to communicate with such systems.
    - ✧ There is no description of the mechanisms of authentication, logging, and access control, which are critical when working with personal data.
    - ✧ Many HRM platforms have closed APIs or complex data formats.
  - Integration into real corporate HR systems may require a complete refactoring of the API architecture, support for OAuth2.0, and Data Privacy control (for example, GDPR or Ukrainian PD legislation).

Table 30. Generalization

Aspect	Potential problem or limitation
Vector Search Scaling	PostgreSQL + pgvector do not scale well without FAISS or Milvus.
Additional training	Requires complex MLOps to synchronise base and model
Interface	Gradio is not suitable for production and enterprise integration.
Language adaptation	Potentially poor quality of Ukrainian- and Russian-language data
Integration with HRM	Missing APIs, authorisation, access logic, and format support

Recommendations for future research:

- For scaling, consider implementing FAISS or Milvus with a separate API.
- For production integration, switch to FastAPI + React/Vue interface.
- For multilingual support – additional training on Ukrainian vacancies + replacement of the model with XLM-R or LaBSE.
- For integration with HRM – development of REST API with authorisation (OAuth2 / JWT), support for JSON Resume / LinkedIn schemas.
- For stable updates, CI/CD and pipelines are implemented to update the model + database recording.

User Manual

The product name Recruiter Assistant is an innovative tool for selecting candidates based on semantic similarities between resumes and job descriptions.

Prerequisites. Before using the system, you must make sure that the following components are installed on your computer:

- Python 3.9+;
- Docker and Docker Compose;
- Installed pip to manage Python dependencies;
- Browser for interacting with the interface (Chrome/Firefox);
- Optionally, use Git to clone the repository.

Steps to prepare for work:

- Clone Repository:

```
git clone https://github.com/DianaVyshotravka/RecruiterAssistant.git
cd RecruiterAssistant
```

- Run a database in Docker:

```
cd backend/db
docker start pgvector-db
```

- Install Python dependencies:

```
pip install -r requirements.txt
```

- Launch the Gradio API and interface:

```
jupyter notebook back.ipynb
```

- The browser will open the Gradio interface (or available at http://localhost:7860)

Data Entry:

- Select the type of source (text or link).
- Enter or upload the relevant information.
- Click Get embedding.
- Select the desired number of candidates and click Find Candidates.

Processing:

- The system pre-processes the text.
- With the help of a neural network, a vector representation is generated.
- Data is stored in the database.
- With the cosine distance, the specified number of candidates is located and displayed.

Getting results:

- At the output, the user will receive a list of relevant candidates.
- It is possible to provide feedback, which in the future can be used for additional system training.

Additional notes:

- The English language of the input text is supported (the system does not guarantee quality results in other languages);
- For greater model accuracy, it is advisable to present structured texts.
- By default, the database is deployed locally and is not public. For teamwork, you need to set up network access.

Repository Link: https://github.com/DianaVyshotravka/RecruiterAssistant

4.7. Performing a Control Example

Interaction through the user interface. Input Format: Text

Since the system provides two types of input, we will select the appropriate input data to check each of them: a random vacancy from the test dataset is selected for the text format. Let's start testing the vacancy in text format. Select the Input type 'from the text' and insert the vacancy text from the test dataset.

Fig.25. Adding the text of the vacancy

After clicking the 'Get Embedding' button, the text processing and saving to the database began, which took up to 3 seconds.

Fig.26. Text processing

We see that the vacancy is encoded and saved in the database with a unique identifier 1. We can check this directly by executing an SQL query.

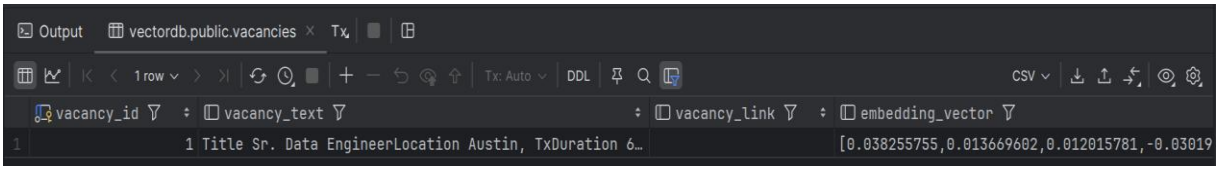

Fig.27. Saved vacancy in the vacancies table

Let's go back to the interface and take the following steps to get candidates. Select the desired number of candidates (e.g. 3) and click the 'Find Candidates' button. After that, the search results will be displayed with the similarity level and buttons for rating.

The system identified this candidate as the most relevant. Although at first glance, it does not seem so because the title indicates that he is a sales specialist, if you read the experience and skills of the candidate, he has significant experience with data and visualisation, as well as skills in using the appropriate tools. Here, the system behaved exactly as we expected it to be – to notice details that a person would ignore and not to judge from the first sentence, but to generalise and abstract. Of course, the data will be reviewed by recruiters later, and this candidate may not be approved for this position, but his resume will be considered.

The following candidates seem relevant even at first glance since they are data engineers or analysts with considerable experience and knowledge of tools and techniques.

After receiving the results, they can be evaluated using the Fit/No Fit buttons, which are located under each resume of the selected candidate (Fig. 28-30). The results can be saved by clicking on the 'Save Feedback' button. For the experiment, 3 candidates were marked as 'No Fit' and all the others as 'Fit', and the results were saved.

Let's also check the fact of recording user ratings in the database directly. To do this, run an SQL query to retrieve the contents of the feedback table.

Fig.28. First potential candidate

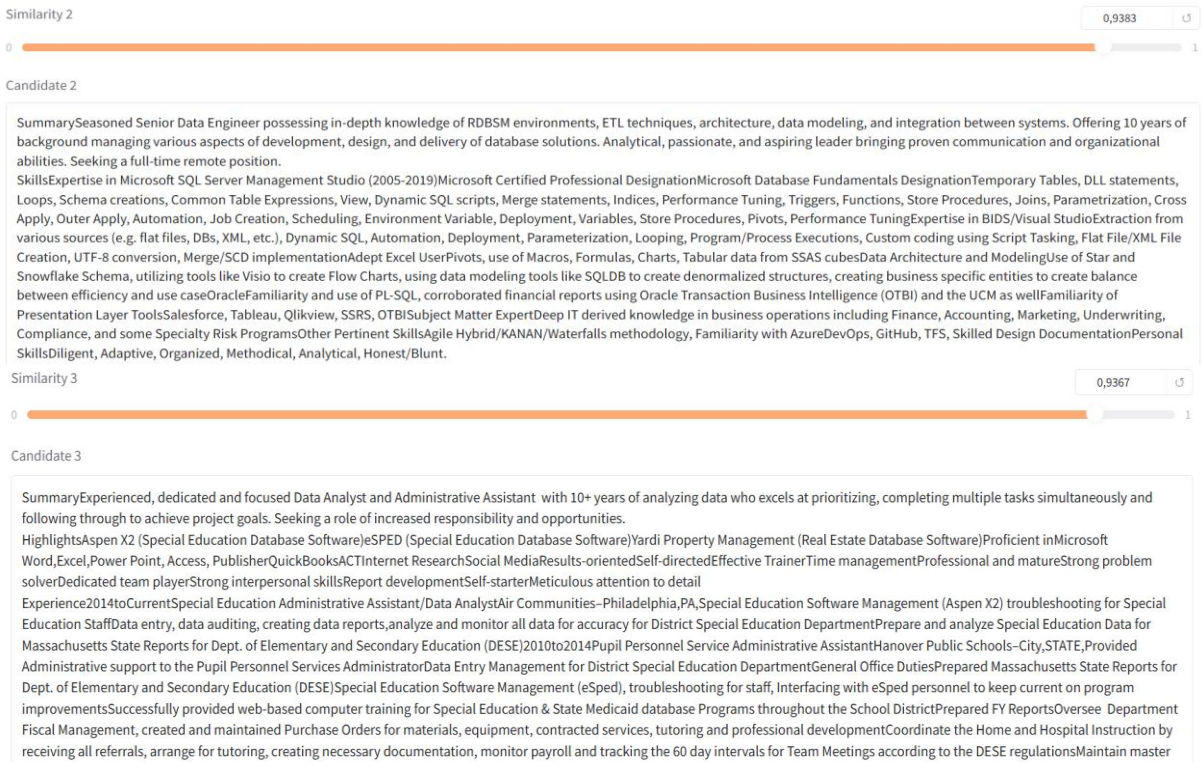

Fig.29. Resume fragments of the following two candidates

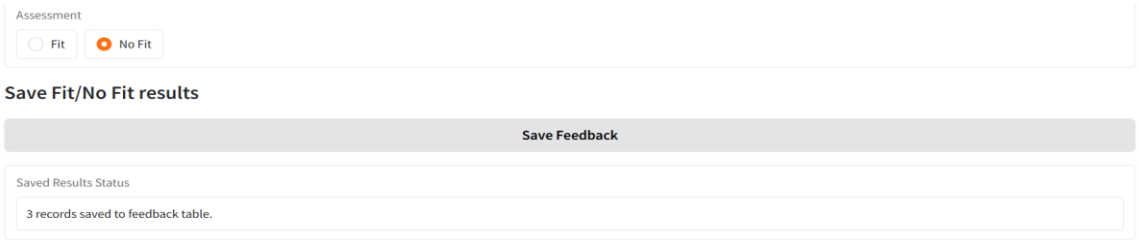

Fig.30. Saving feedback

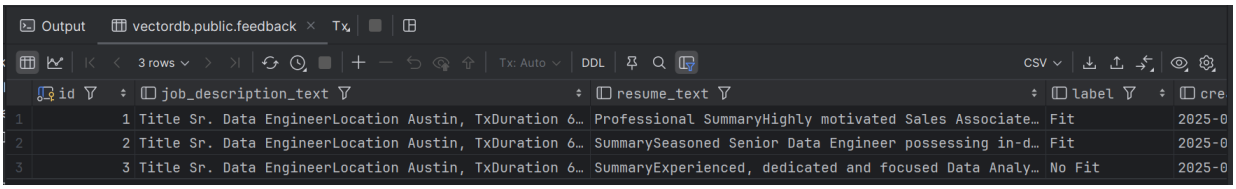

Fig.31. Content feedback after saving reviews

We can see that all three records are indeed saved in the table with the ratings Fit and No Fit, respectively. The format of the table is almost identical to the training dataset and, after accumulating enough data, will be used to train the model further with reinforcement.

Interaction through the user interface. Input format: links.

To test the selection of candidates for a vacancy in the link format, an active vacancy that develops today https://jobs.develops.today/jobs/full-stack-javascript-developer was selected. After selecting the input data, we will start searching for candidates. Select the Input type 'from URL' and insert the link chosen for testing. After that, click the 'Get embedding' button to process the link and vectorize the text.

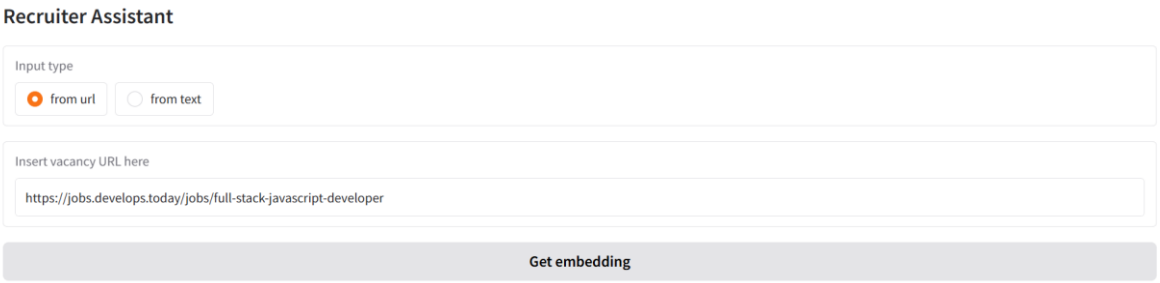

Fig.32. Entering a vacancy at the link

As a result, we get the cleaned text of the vacancy obtained as a result of parsing (Cleaned text field). We also see a message about the success of data embedding and the corresponding unique identifier of the vacancy in the database.

Similar to the previous test example, let's check the contents of the vacancies table of a vector database using an SQL query. As you can see from the result of the query (Fig. 34), in addition to columns vacancy_text and embedding_vector, the vacancy_link field is also filled in.

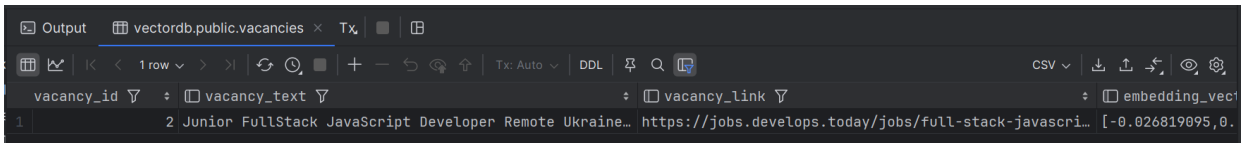

Fig.33. The vacancy is saved in the vacancies table

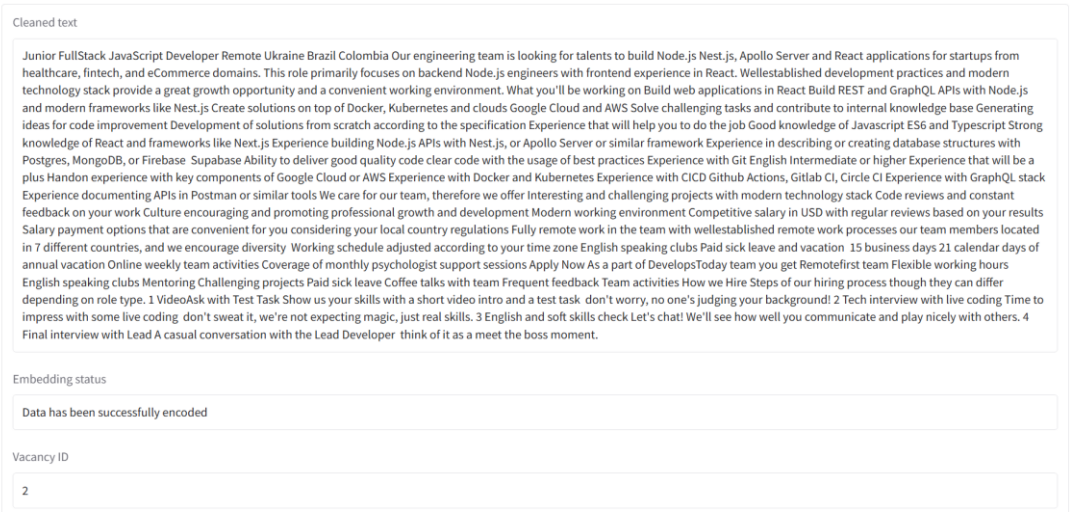

Fig.34. The result of web page parsing and text embedding

Let's go directly to the search for candidates. To do this, again, select the number of candidates we want to receive (let it be 2) and click the 'Find Candidates' button.

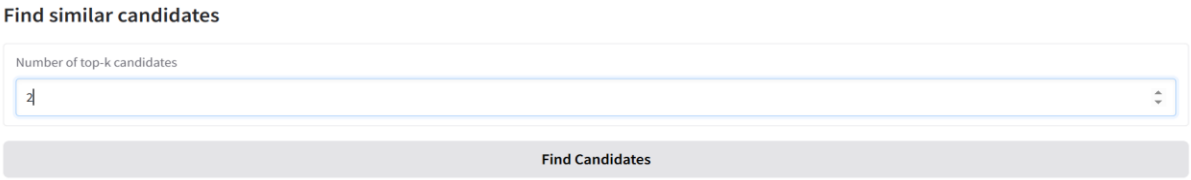

Fig.35. Request to search for relevant candidates

As a result of the request, we will receive the text of the vacancies of 2 candidates and an assessment of their similarity.

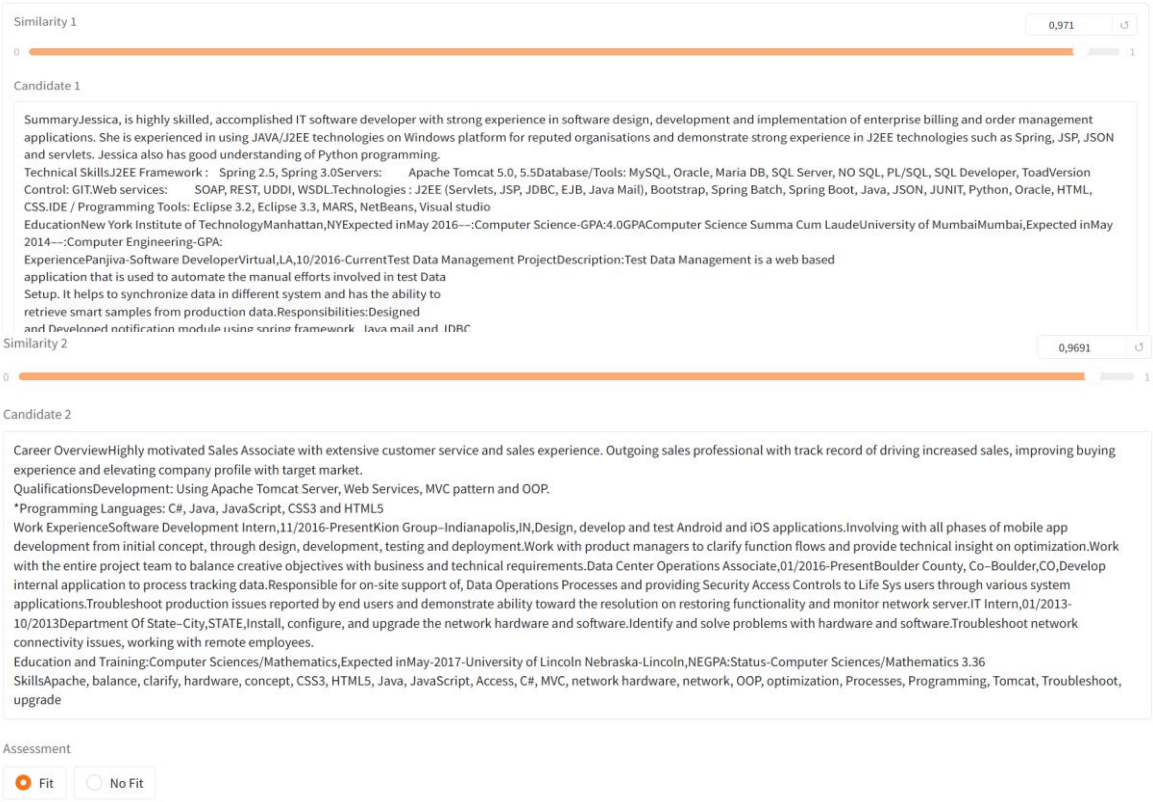

Fig.36. Results of relevant candidates returned by the system

As you can see, both candidates know programming patterns, have experience in web application development, have relevant technical education, and are familiar with the tools required. Again, candidates selected by the system are expected to be re-reviewed by recruiters. After evaluating the results, you can leave your feedback and save it in the database by clicking the 'Save Feedback' button.

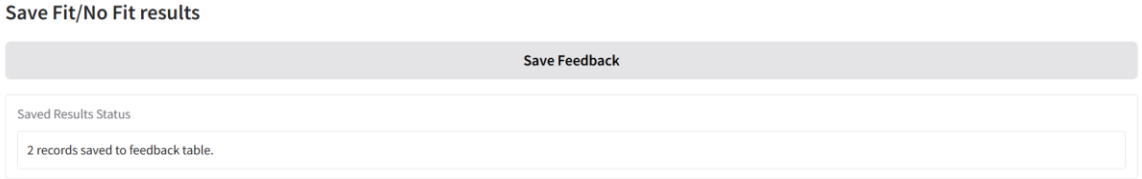

Fig.37. Saving feedback to the database

Similarly, we will check the results of saving reviews by executing an SQL query directly into the database.

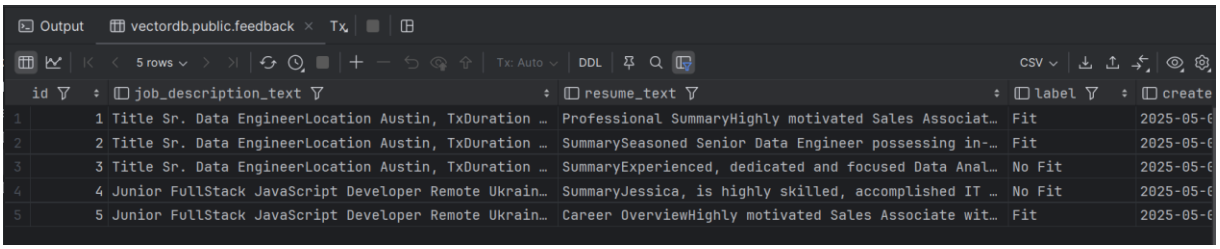

Fig.38. The contents of the feedback table after saving new records

As you can see, two new queries have been added with the corresponding description of the vacancy, the proposed candidates and the user's rating.

Demonstration of the Vector Database

A rather important element of the system is the vector database because it selects candidates. Before that, we checked its functioning as a regular database using examples of saving records, and now we will check for the search for similar vectors.

A query is made to calculate the cosine-bearing distance between the generated vector of the corresponding size (1.768) and the candidate embeddings. Since such an operation calculates the distance, and we need to display the similarity, we will calculate it as 1-cos_distance since the vectors are pre-normalised. Let's display only the first 10 results.

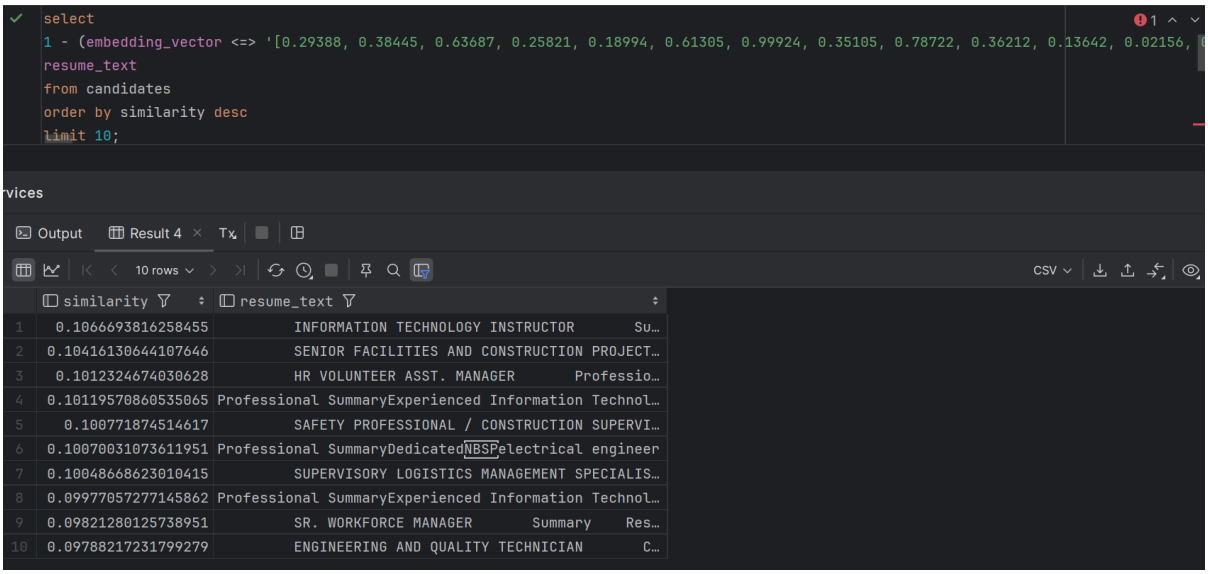

Fig.39. Request and its result

As you can see, the similarity column and the vacancies text have been returned. The low level of similarity is explained by the fact that the vector was generated by chance and can carry a really irrelevant vacancy relative to the candidates. With the help of a nested query, a query was also made to search for relevant candidates for the vacancy with id = 1 (the same one used in the first test case).

As expected, the first three candidates coincide with those that were returned to the user through the interface. Therefore, the database and interface work smoothly.

4.8. Analysis of the Results Obtained

Analysis of the results of the tested models

Since an experimental approach was used to select the models that will form the basis of the embedding model, we will review the tested models and their results. Three models were built based on BERT, RoBERTa and SimSCE RoBERTa (sentence transformer). For each of the models, the same loss function was used – CosineEmbeddingLoss and the AdamW optimiser.

BERT (Bidirectional Encoder Representations from Transformers) is a basic transformer model from Google that was trained on large corpora of English using bidirectional context. It means that it takes into account both the preceding and subsequent words when constructing a contextual representation. The results of model training are presented in Fig. 41 after training for five epochs.

```
select
1 - (embedding_vector <=> (select embedding_vector from vacancies where vacancy_id = 1)) as similarity,
resume_text
from candidates
order by similarity desc
limit 10;
```

rvices

Output | Result 5 | Tx | 10 rows

	similarity	resume_text
1	0.9482989593833379	Professional SummaryHighly motivated Sales Associat…
2	0.9382728934288025	SummarySeasoned Senior Data Engineer possessing in-…
3	0.9367117844052203	SummaryExperienced, dedicated and focused Data Anal…
4	0.9344858527183533	Professional SummaryOver 7 years of progressive hum…
5	0.9341674486298907	Professional Summary5 years of experience on Datab…
6	0.9330844283103943	SummaryIncredibly motivated professional with a var…
7	0.9330118338458654	Professional Summary5 years of experience on Datab…
8	0.93258136510849	Career OverviewHighly accomplished Data Architect /…
9	0.9315562844276428	Career OverviewClients/Employers: Thompson Reuters …

Fig.40. The result of the search for candidates for the vacancy with id = 1

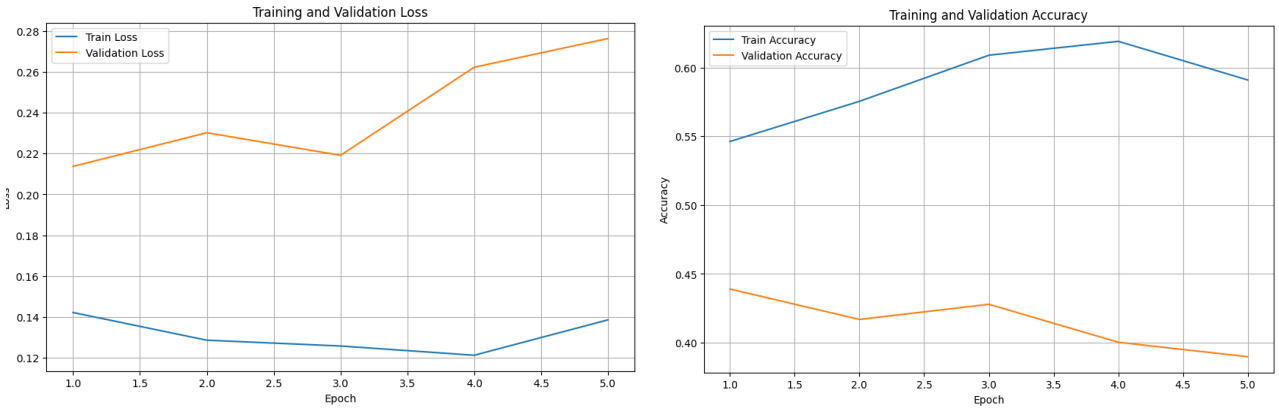

Fig.41. Learning outcomes of the model based on BERT

In general, the result is not very bad, but the model is slightly overtrained, and over the centuries, the accuracy decreases.

RoBERTa (Robustly Optimised BERT Pretraining Approach) is an improved version of BERT developed by Facebook AI that has been trained for longer, on a larger amount of data, and without the use of next-sentence masking (NSP). It retains the BERT architecture but is more effective thanks to a modified training strategy. It can be seen in Fig. 42. Although the losses during the epochs are greater than in BERT, the graph is much more stable, there is no retraining, and the accuracy is higher compared to the previous model.

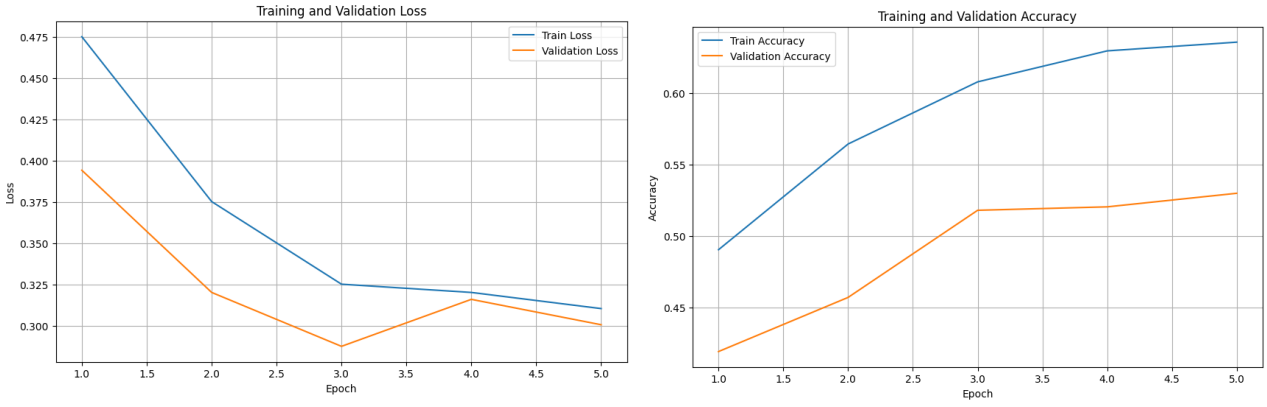

Fig.42. Result of training a model based on RoBERTa

Last in line, but not according to the results, is the model that SimSCE RoBERTa tested. It is a version of RoBERTa, further trained using the Supervised SimCSE technique, an approach that explicitly optimises the model for building high-quality sentence embeddings. SimCSE is trained in such a way that similar sentences have close vectors, and dissimilar ones have distant vectors. Such a model is ideally suited to the problem that is being solved and has demonstrated the best result (Fig. 43).

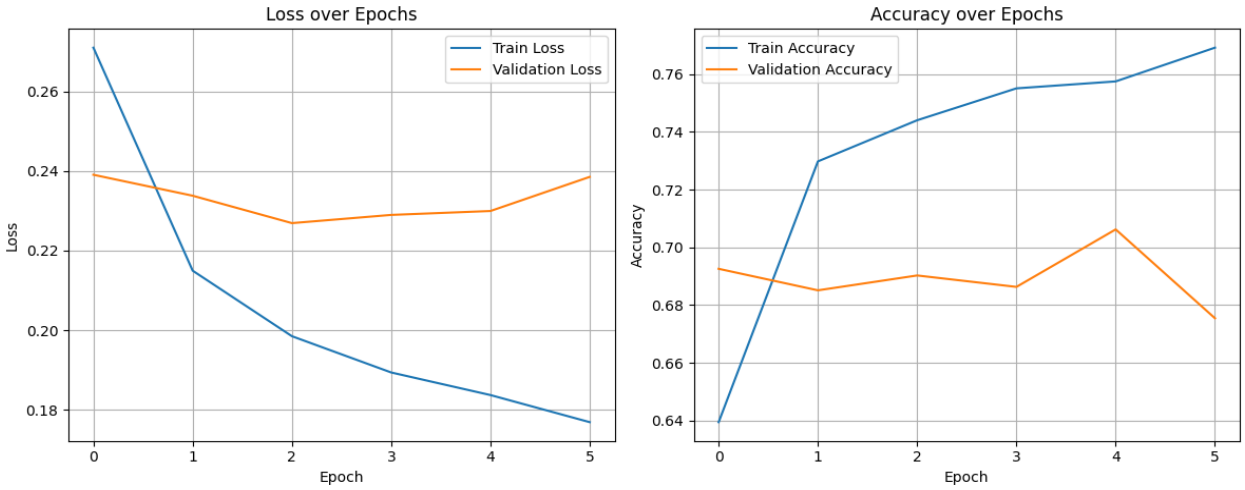

Fig.43. Learning outcomes of a model based on SimSCE RoBERTa

So, the best model turned out to be based on the sentence transformer SimSCE RoBERTa, which is quite an expected result.

Analysis of the results of the application of various loss functions

The selected loss function has a significant impact on the model results, so several such functions were additionally tested: TripletMarginLoss, CosineEmbeddingLoss, MSEloss, and InfoNCE.

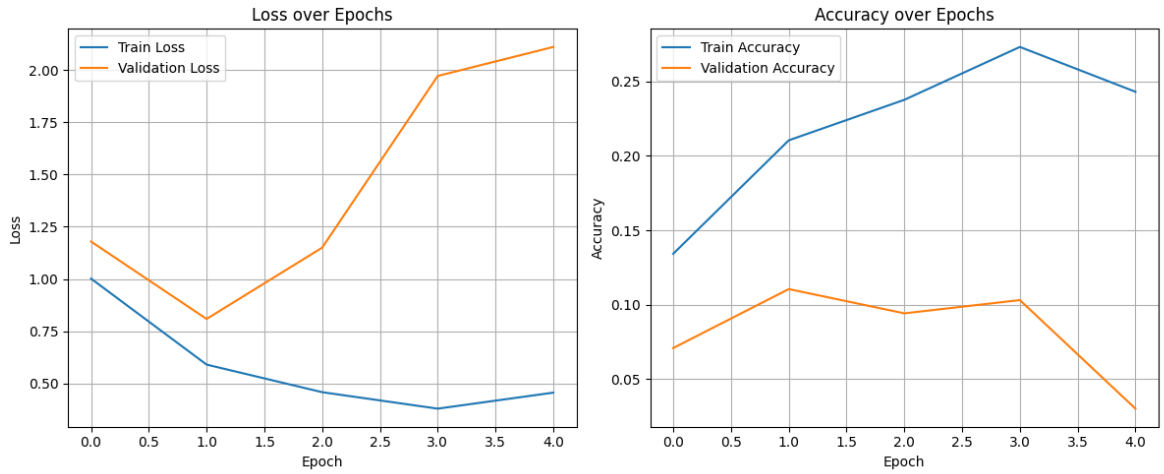

Fig.44. RoBERTa + TripletMarginLoss

The TripletMarginLoss function is used to train models in such a way that positive examples (similar pairs) are closer in vector space to the anchor than negative (dissimilar examples) with a certain margin. It requires that each dataset record contain an anchor (vacancy), a positive pair (candidate, where label == Good Fit) and a negative pair. Accordingly, a dataset consisting of triplets is formed to use this function. It was applied to the RoBERTa-based model, and the results can be seen in the graph (Fig. 44).

We can note a considerable loss, which is initially expected, since the approach to working with data triples cannot immediately give a positive result, but with epochs, losses and accuracy do not improve, and there is even a specific process of retraining.

CosineEmbeddingLoss is used to optimise cosine similarity between vectors. If the label is 1, then the model tries to make the vectors as close as possible (cosine similarity closer to 1), and if -1, then more distant (cosine closer to -1 or 0). This loss function was used in tandem with each of the three models (BERT, RoBERTa, and SimSCE RoBERTa), Fig. 44 – 46, respectively, and showed quite good results on average.

MSELoss is a classic loss function for regression that calculates the square of the difference between the predicted value and the actual value. Despite the initial purpose, it showed the best result among the lost functions applied. It was tested in tandem with SimSCE RoBERTa. The results can be evaluated in Fig.45.

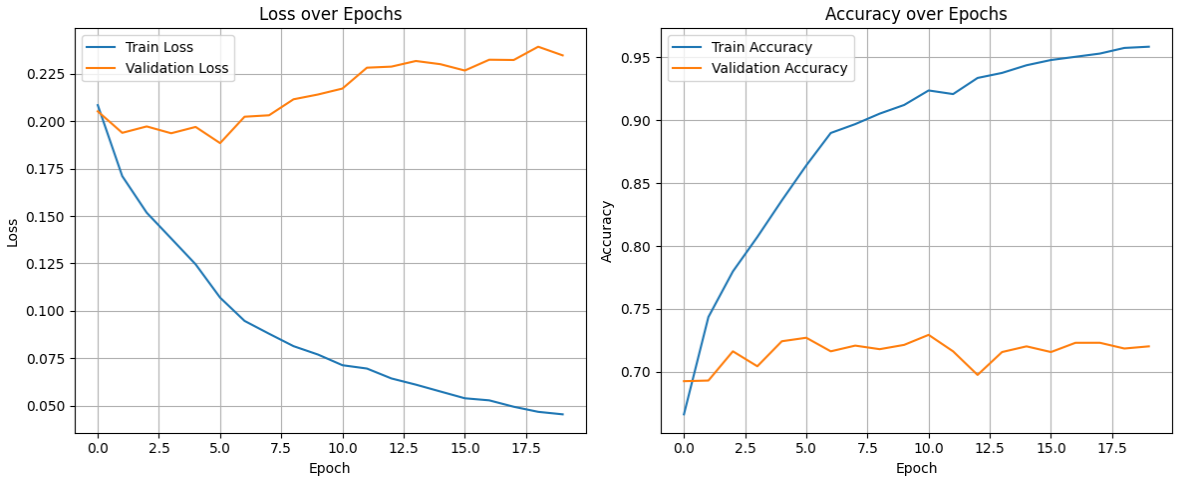

Fig.45. SimSCE RoBERTa + MSELoss

Although the graph shows obvious retraining, it was possible to obtain the final model with an accuracy of 73% on validation data, which is the best result obtained.

InfoNCE is a contrastive loss function that is widely used in self-supervised learning, in particular in methods such as SimCLR, SimCSE, etc. It optimises the model so that positive pairs are closer to each other, and everyone else from the batch is considered harmful. This function also required special data. In particular, only positive pairs were transferred, and the loss function considered all other candidates to be negative examples for each positive batch pair. The learning process and results are somewhat similar to the example using the TripletMarginLoss function.

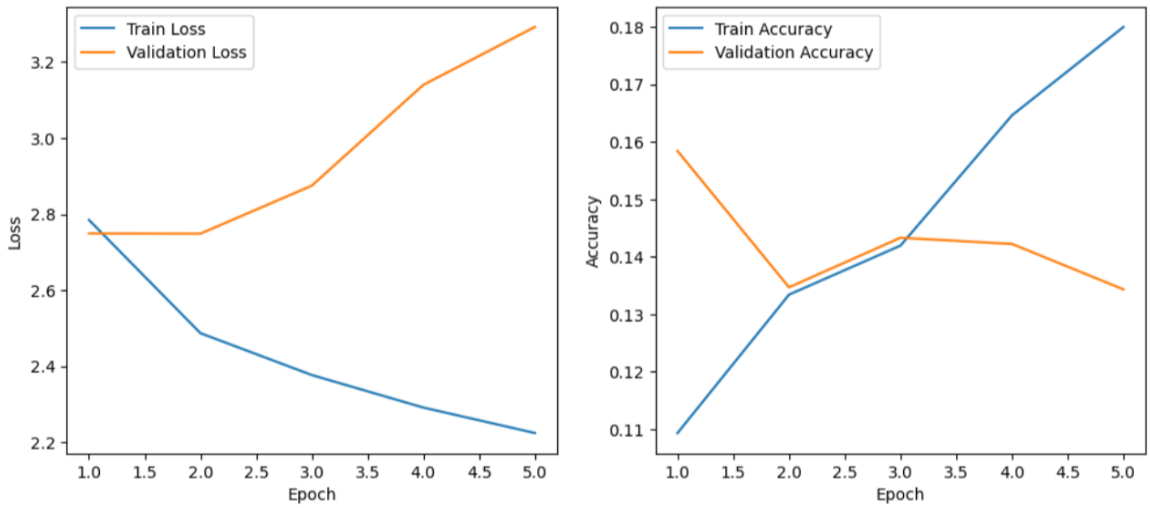

Fig.46. SimSCE RoBERTa + InfoNCE Loss

Thus, the best result was given by the MSE loss function in combination with a model based on SimSCE RoBERTa, and this model was applied to the system. Such a difference in the results of most loss functions with those used by triplets is most likely caused by the fact that the dataset initially contained positive and negative pairs in the exact equivalent. Therefore, the explicit additional creation of such pairs did not improve the result.

Analysis of the results of the final model and system operation

As mentioned earlier, the assessment of embedding models is carried out on a specific task. In the context of the problem to be solved, a classification task is chosen for evaluation. Thus, the model is evaluated.

	precision	recall	f1-score	support
0	0.7405	0.6628	0.6995	857
1	0.7087	0.7794	0.7423	902
accuracy			0.7226	1759
macro avg	0.7246	0.7211	0.7209	1759
weighted avg	0.7242	0.7226	0.7215	1759

Fig.47. Evaluation of model accuracy

The model achieved an accuracy of 72% on the classification problem, and the model is not biased against classes. Additionally, the Precision at 10 metrics (0.7543) was calculated for the recommendation system problem. It measures the proportion of relevant items among the top 10 results returned by the system. In other words, this metric shows how much of the first 10 outcomes offered by the system are really relevant.

Fig.48. Metric Precision @ 10

This accuracy shows that out of 10 returned candidates, seven will be relevant, which is a pretty good result for a recommendation system, given that additional training is possible on user reviews. Precision@10 is a metric that shows what proportion of the first 10 recommended candidates the system really selected correctly (i.e. these candidates really meet the requirements of the vacancy or would be useful to the recruiter). Formally:

$$Precision@10 = \frac{\text{Number of relevant candidates among the first 10 recommendations}}{10}.$$

It calculates accuracy only among the top 10 positions on the recommendation list, not among all those whom the system could recommend. Suppose the system recommended 10 candidates, and of these:

- 7 Candidates are really good for the job (relevant)
- 3 – not quite compliant

Then $Precision@10 = \frac{7}{10} = 0.7$. It means that 70% of recommendations in the top 10 were helpful. In the field of recruitment, Precision@10 is important because the recruiter most often looks only at the first few candidates. Therefore, this metric helps to assess how "strong" the top recommendations of the system are, and not the average level of the entire base. To estimate the response time of the system, the interface code has been slightly modified to calculate the time spent performing basic functions. Thus, for the task of finding candidates for a vacancy from the text, time is needed:

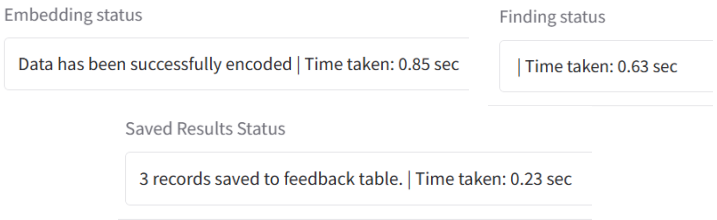

Fig.49. Time to perform key functions (input type: text)

Similarly, the time for searching for candidates for a vacancy is fixed at the link:

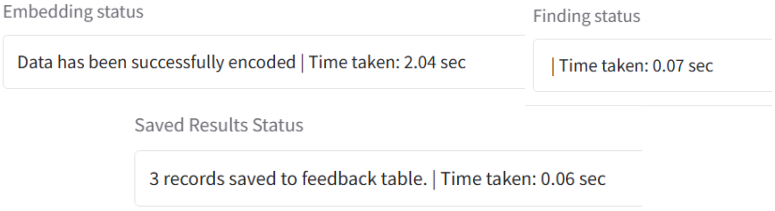

Fig.50. Time to perform key functions (input type: url)

We can see that, in general, the system requires much less time for key functions than indicated in the requirements (up to 30 seconds). Most of the time is spent on embedding, especially if it involves parsing a web page, which is the expected result. However, in general, the system works quite quickly. For clarity, let's summarise the results achieved

from the list of initially defined tabular requirements in tabular format.

Table 31. Analysis of the implemented functionality

Requirement Type	Requirement	Implemented
Interoperability	The ability to read information from resumes available on web resources at the link using parsing.	**Fully implemented**
	It is possible to accept resumes in text format as well.	**Fully implemented**
Mobility	Full availability from any device through a browser. No need to install additional software	**Partially implemented**. Since the website is not public, you need to download the model and run the file. After a full release, the system can be accessed via the link.
Functionality	Taking into account the content of the vacancy and implicit connections in the text (synonyms, paraphrasing, professional vocabulary).	**Fully implemented**, the accuracy of the model exceeds 75%, which guarantees approximately 7/10 of relevantly selected candidates.
	Automatic output of the most relevant candidates by vector similarity.	**Fully implemented**
	Interpretation of why exactly these candidates are suitable (highlighting key coincidences, semantic correspondences).	**Not Implemented**
Usability	Intuitive UI with hints, Visual indication of relevance	**Fully implemented**
Performance	Response time to candidate search: up to 60 seconds. Job encoding time: less than 30 seconds.	**Fully implemented**, any system functionality takes up to 5 seconds.
Reliability	Redundancy, fault tolerance, and automatic model recovery	**Not Implemented**
Supportability	CI/CD pipeline for updating models and monitoring the accuracy of models in a production environment.	**Not Implemented**
Additional requirements: Active Learning	Collecting feedback from the recruiter about the relevance of the results.	**Fully implemented**
	Periodic additional training of the model on this feedback.	**Partially implemented**: The model is not difficult to train on new data. Only a sufficient amount of it is needed; reinforcement learning is used. You need to create a new model.

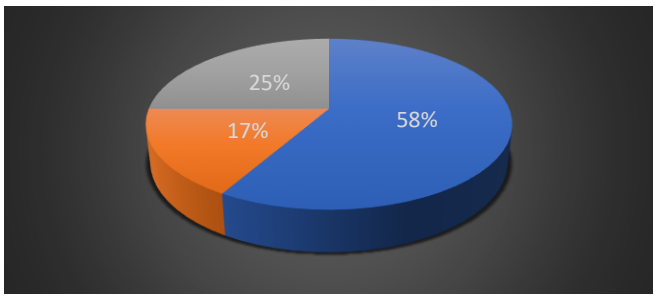

Fig.51. Visualisation of the implemented requirements of the system functionality, where blue is fully implemented, orange is partially implemented, and grey is not implemented

Most of the requirements that are not implemented relate to the global deployment of the system. Since it works locally as a demo version, there is no need to create pipelines to update the model or the ability to access from any device.

Contextual comparison of model results

The study obtained an Accuracy is 72% and a Precision@10 is 75%. These results look positive at first glance, but without a clear comparison with basic systems or previous work, they remain out of context. The following is an analytical framework for the comparison:

- Comparison with basic models (Baselines). In typical systems of classification of pairs "resume-vacancy", the following basic approaches are used:
  - TF-IDF + Cosine Similarity:
    - ✧ Without a deep understanding of semantics;
    - ✧ Expected accuracy – 50–58%;
    - ✧ Precision@10 – often below 60% because it is based on the coincidence of words, not the meaning.
  - Bag-of-Words + Logistic Regression:
    - ✧ Poor generalizability, high sensitivity to formulations;
    - ✧ Accuracy – ~60%;

- ✧ Precision@10 – 55–65%;

- Sentence-BERT or Universal Sentence Encoder (without additional training):

  - ✧ Using ready-made sentence embeddings;
  - ✧ Accuracy – 65–69%;
  - ✧ Precision@10 - up to 70% in the best conditions.

The indicators of the system in the article exceed the traditional basic approaches by at least 3–7%, mainly due to additional training on specific pairs (Good Fit / No Fit).

- Comparison with similar scientific systems

  - Job2Vec (LinkedIn Engineering) [37-45]:

    - ✧ Specialised vector submission of vacancies and resumes;
    - ✧ Precision@10 ≈ 70%;
    - ✧ Recall@10 ≈ 60%;
    - ✧ Control relevance with actual click or hiring data.

  - Deep Semantic Matching Network for Job Matching [46-57]:

    - ✧ Uses attention and job-aware representations:
    - ✧ Classification accuracy ≈ 74–76%;
    - ✧ Precision@10 ≈ 73–77%.

The results of the study are at or slightly below the best scientific models. Still, those models are usually trained on their large datasets with careful human markup or actual hiring data, which is not the case in this study.

For a better interpretation of performance indicators, comparisons should be made with basic and modern approaches such as TF-IDF + cosine similarity, Sentence-BERT and attention-based models. The reported accuracy scores (72%) and Precision@10 (75%) indicate improvements compared to traditional approaches, but remain slightly lower than the best results obtained from proprietary corporate data.

To evaluate the effectiveness of the proposed approach (SimCSE-RoBERTa + Siamese network), an experimental comparison was carried out with the classical vectorisation and classification methods TF-IDF + SVM, BM25 and BERT (without additional training, with cosine similarity). The model was evaluated on the "relevant/irrelevant" classification task with Accuracy, Precision, Recall, F1-score, and Precision@10 metrics (for the top 10 candidates).

Table 32. Comparison with basic approaches

Method	Accuracy	Precision	Recall	F1-score	Precision@10
TF-IDF + SVM	61%	58%	65%	61%	50%
BM25	64%	60%	68%	64%	55%
BERT (cosine similarity)	68%	65%	70%	67%	62%
SimCSE-RoBERTa + Siamese	**72%**	**70%**	**74%**	**72%**	**75%**

As can be seen from the table, the proposed approach showed better results in all metrics, mainly due to better consideration of context and semantics. Here is a graph illustrating a comparison of the proposed approach with basic methods on five metrics.

To evaluate the effectiveness of the proposed approach (SimCSE-RoBERTa with Siamese network architecture), an experimental comparison was carried out with the basic methods: TF-IDF in combination with SVM, BM25 and the basic BERT model, which was used without additional training to calculate cosine similarity. As can be seen from the graph, the proposed method significantly exceeds traditional approaches in all key metrics:

- Accuracy increased to 72%, which is 4% higher than BERT and 11%–12% higher than TF-IDF and BM25.
- Precision and Recall also showed an increase: 70% and 74%, respectively.
- The F1-score, which reflects the balance between precision and recall, has increased to 72%.
- The Precision@10 indicator is especially indicative: among the top 10 candidates, the proposed method returns an average of 7-8 really relevant resumes (75%), which is much better compared to the basic techniques (50-62%).

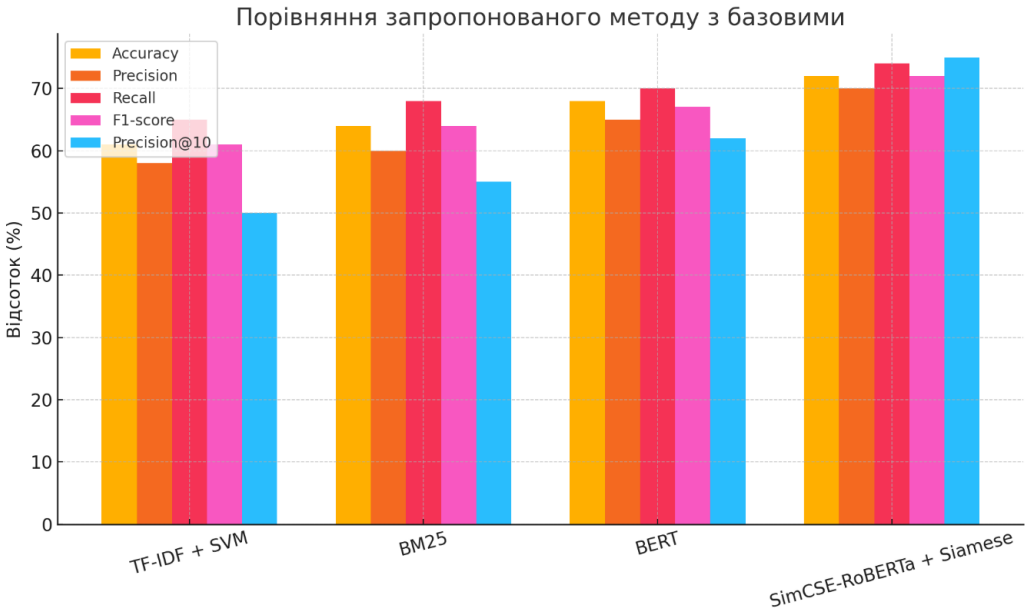

Fig.52. Comparison of the proposed approach with basic methods on five metrics

These results demonstrate the advantages of an in-depth semantic approach, which allows the model to take into account context, synonymy, and atypical wording in resumes and job vacancies. In contrast, classical methods remain word-sensitive and less able to detect hidden connections between a candidate's experience and job requirements. Thus, the use of SimCSE-RoBERTa with the Siamese architecture can significantly increase the relevance of candidate selection and the quality of recommendations compared to traditional text approaches.

Siamese Transformer Architecture

Siamese network is a special model that consists of two identical branches (encoders) with shared weights. In our case, these branches are built on the basis of a transformer model (for example, RoBERTa or SimCSE-RoBERTa). How it works:

- One branch (coder) processes the summary text.
- Another branch is the text of the vacancy.
- Both texts are converted into vectors in one common semantic space.
- The two vectors are then compared to each other, usually using cosine similarity or another metric.

Training takes place in such a way that:

- Relevant pairs ("resume-vacancy") were located close in vector space.
- Irrelevant couples are far from each other.

Thanks to standard weights, both coders learn to "see" texts in the same way and compare semantic similarities, and not just coincidences of words. A key advantage is that Siamese architecture allows the model to learn how to assess the similarities between two different texts (resume and vacancy) in terms of content, not just keywords.

As for the overall values for the model (using the macro average across classes): Overall Precision – 0.7246, Overall Recall – 0.7211 and Overall F1-Score – 0.7209.

Reasons for Performance Degradation for Niche Roles:

- The lack of positive examples in the training sample → model does not form a sufficient vector cluster.
- Semantic ambiguity of names/terms (for example: "Quant Analyst" vs "Risk Modeller").
- High level of "semantic sparsity": even similar resumes can be written with very different wording.
- Domain mismatch – the model was trained in massive, not highly specialised positions.

Solution: Terms of Reference (TOR)implementation of multi-attempt training (Few-shot learning). Here's how you can improve the quality of selection for rare vacancies:

- Few-shot fine-tuning:
    - Additional training of the model on a couple of dozen examples of vacancies/resumes from a specific niche.

- Use of contrastive learning methods (SimCSE, Triplet Loss) even on small sets (10–50 pairs).
- Suitable when the company wants to recruit for specific positions (e.g. only pharmacists or defence R&D).

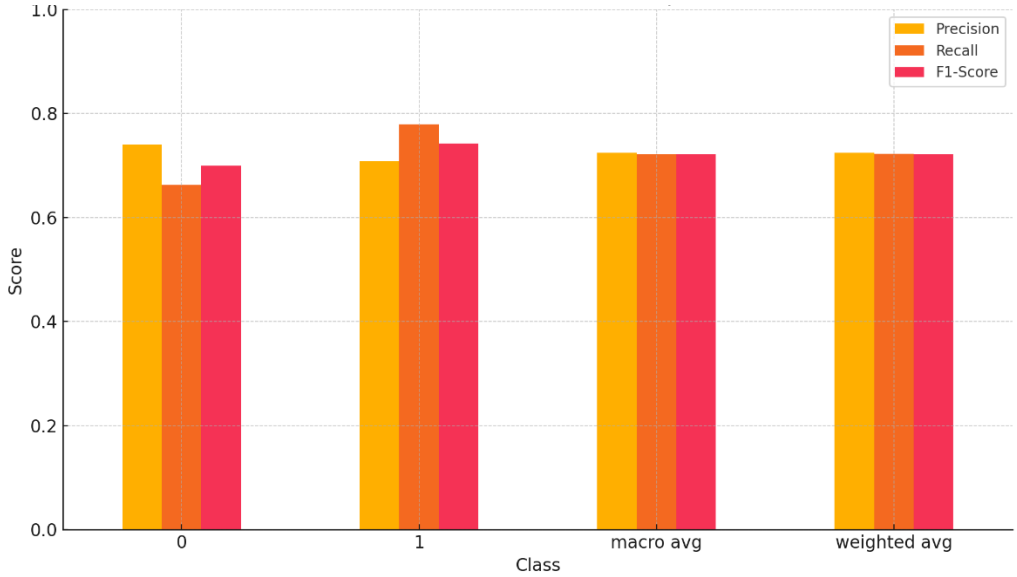

Fig.53. The chart for Precision, Recall, and F1-Score for each class and the averages

- Embedding adaptation / Prompt-tuning. It is not a complete retraining but an adaptation through additional "hints" to the model (for example, we add the context of the vacancy as a prompt). For example: "This is a resume for a narrow role in cybersecurity (penetration testing): ..."
- Hard negative mining. Even with a small number of positive examples, it is possible to generate complex negative pairs that are similar but not relevant. It enhances the model's ability to "distinguish" boundaries in a narrow domain.
- Active learning + clustering. It is necessary to use existing vectors to build clusters of niche candidates. Then ask the recruiter: "Is this candidate suitable or not?" → Collect grades for additional local training.

Table 33. Recommended implementation in the system

Component	Solution
Interface	Add "niche fine-tuning" mode: upload CSV/JSON with 10-100 resumes and vacancies.
Model	Use SimCSE-RoBERTa or sentence-tuned XLM-R with support for further fine-tuning.
Base	Create a separate subcluster in the vector DB for each niche category.
Evaluation	Add metrics for small sets: Precision@k, MRR, Clustering purity.
Adaptation	Support for the constant collection of new examples through user feedback (active learning loop)

To maintain high-quality selection in narrow niches, the system must implement a few-shot learning scenario where:

- A user or client can provide several examples,
- The system further trains its layer or vector behaviour,
- And stores a separate adapted "industry logic" for specific domains.

Let's describe a complete plan for fine-tuning niche specialisations (using the example of Data Scientist, pharmacists, and UI/UX designers) and architectural terms of reference (TOR) for the few-shot adaptation module for the Recruiter-Assistant system. The goal is to improve the relevance of candidate selection in narrow niches (<100 examples), through local retraining of the SimCSE transformer or another sentence embedding model – name of the Few-shot adaptation module: few_shot_finetuner – sentence embedding model subsystem on small datasets.

Safety:

- User data is not saved after fine-tuning (auto-purge option).
- Isolation of models by segments (UI/UX, pharmacists, etc.).
- Optional pre-workout/inference authorisation.

Table 34. Examples of niche specialisations

Component	Description
Data Scientists	
Data	50–100 pairs "vacancy ↔ relevant resume" + 50 pairs "↔ vacancy irrelevant resume"
Format	CSV/JSON with fields job_text, resume_text, label (1/0)
Test Kit	10 pairs for validation (not included in the train)
Basic model	sentence-transformers/paraphrase-MiniLM-L6-v2 or SimCSE-RoBERTa
Training	Fine-tuning with **contrastive loss** (triplet or cosine) at 2–5 epochs
Exodus	The new simcse-finetuned-datasci model is → loaded as a separate pipeline.
Pharmacists	
Data	Less than 50 examples → generate negative pairs automatically (Hard Negative Sampling)
Model	Support for biomedical terms is necessary → preferably use biobert-base-cased as a basis.
Fine-tuning	Few-shot scenario: epochs = 5, learning rate = 2e-5, batch size = 8
Purpose	Increase recall for key skills: GMP, pharmacology, and prescription software.
UI/UX designers	
Semantic discrepancies	"UI designer", "product designer", and "interaction architect" have blurred borders.
Data	60 examples of "resume vacancy ↔" (manual markup from a recruiter)
Feature	The vital role of the portfolio (not just text) — add an explanation via Prompt: "The following is a resume of a UX designer"
Adaptation	The model is stored as simcse-finetuned-uxdesign + a separate vector index.

Table 35. Functionality

Function	Description
upload_dataset()	Upload CSV/JSON with examples (vacancy + resume + tag)
train_model()	Fine-tuning sentence-transformer (SimCSE, MiniLM, BioBERT)
evaluate_model()	Calculation of Precision@k, MRR, and cosine similarity metrics
export_embeddings()	Generate vectors for all resumes in a new domain
register_model()	Saving the model in HuggingFace-style format
load_finetuned_model()	Using an adapted model when searching

Table 36. Technical components

Component	Technology / Libraries
User Interface	Gradio/Streamlit download panel, status view
Core model	sentence-transformers / SimCSE, transformers
Vector Base	PostgreSQL + pgvector or FAISS
Training Pipeline	PyTorch/Trainer API (or SentenceTransformer.train())
Saving models	HuggingFace-style checkpoints (.bin, config.json)
Monitoring	WandB (optional), logging in SQLite

Table 37. Challenges and limitations

Aspect	Risks/features
Number of examples	At least 10 pairs for meaningful fine-tuning
Overfitting	High risk → EarlyStopping, Dropout
Inference after training	Should automatically update the vector base index

Conclusion of a complete plan for fine-tuning niche specialisations:

- New pipeline named: simcse-finetuned-{role}-{timestamp};
- Updated vector base: vectors_{role};
- Ready to use through the main search interface in Recruiter-Assistant.

Approximate time for fine-tuning:

- <100 pairs, ~2-5 min per GPU (Colab / local);
- Vector inference – ~1 min for 500 resumes.

5. Conclusions

The idea of the project was prompted by the analysis of the labour market and the work of the HR sphere, which requires manual primary processing of many vacancies and resumes of candidates. Despite the presence of numerous recruiting platforms, existing solutions are primarily focused on keywords rather than a deep semantic correspondence between the requirements of the vacancy and the candidate's experience. It leads to a high level of selection errors, the loss of potentially strong specialists or, conversely, the hiring of unsuitable persons. Thus, the general goal of the project is to simplify and, to some extent, automate recruitment processes, namely the selection of candidates for a vacancy, through the introduction of an intelligent recruiting assistant that uses modern methods of natural language processing, semantic search and machine learning. Such a system is designed to reduce the time that the recruiter spends on processing candidates' resumes while not worsening, but on the contrary, improving the quality of selection. The creation of such a system required the fulfilment of sub-goals. The most expedient method of implementing the selection of candidates has been chosen by comparing alternatives according to 5 criteria: speed of implementation, independence from third-party services, flexibility and customisation, taking into account semantics, and accuracy of selection. Unlike traditional approaches to automatic recruitment, which are based on rules, keywords or heuristic methods, this project implements model training based on the semantic comparison of paired examples. It made it possible to reduce the dependence of the system on superficial coincidences in texts and improve the model's ability to generalise to new, previously unseen examples. In addition, the principles of weak mark-up and positive-negative generation of pairs are used to form the training dataset, which minimises the need for a large amount of manual annotation and increases the scalability of the solution. The requirements for the system under development have been formed.

As a result of the development, a text embedding model based on SimSCE RoBERTa was created in combination with the MSE loss function and the application of the Siamese network approach. Before that, the text of vacancies was analysed for the presence of specific patterns, and subjectivity and polarity were assessed.

To store data and, most importantly, select candidates by the similarity of their embedding vectors, a vector database has been created and filled.

User interaction is made possible by the Gradio API, which defines functions for embedding using a pre-saved model and searching for relevant candidates using direct requests to the database.

Since the initial idea of the system was to improve results through additional training with reinforcement based on user feedback, it is possible to assess the relevance of each candidate returned by the system. The data obtained is stored in the feedback table and can be used in the future for further training of the model.

As a result of this project, a prototype of an intelligent recruiting assistant was implemented, capable of automatically searching for relevant candidates based on a deep semantic analysis of the text of vacancies and resumes, this allows you to solve one of the key problems of modern recruiting – dependence on superficial matching of keywords, which often leads to errors in the selection of personnel.

The developed system is based on modern methods of natural language processing and neural network models, in particular, on the SimSCE RoBERTa model in Siamese architecture. It allows you to effectively compare the texts of vacancies and resumes in a common vector space, which provides better generalisation of the model to new examples and significantly increases the accuracy of recommendations. For this purpose, the following has been implemented:

- Pipeline of preparation and cleaning of resume and vacancy texts, including the identification of patterns, polarity and subjectivity;
- Semantic search based on a vector database in which candidate embeddings are stored;
- User interaction interface via the Gradio API for convenient entry of vacancies and viewing results;
- Feedback collection mechanism for further model training.

At the same time, the boundaries of the implemented version of the system are clearly outlined: it currently functions as a local demonstration and needs further development to work in a scalable online environment, including building CI/CD pipelines, ensuring availability through cloud infrastructure, as well as expanding to other use cases (for example, job recommendations for candidates). Since the MVP has been created, the main emphasis can be placed on improving and expanding the functionality. Consider the following prospects:

- Implement the interpretation of model results to increase user trust in the system.
- Creating a model for reinforcement learning from human feedback to improve model outcomes.
- You can try to improve the results of the model by taking keywords into account and combining the semantic embedding model and the standard TF-IDF model using the cross model.
- Expanding the functionality of the model to work not only for recruiters but also for candidates (technically, this is not difficult to do since you do not need to change either the embedding model or the search logic, but only fill in the vacancy table and modify the logic of the search function for candidates).
- Implementation of pipelines to support and update models.

Acknowledgement

The research was carried out with the grant support of the National Research Fund of Ukraine, "Information system development for automatic detection of misinformation sources and inauthentic behaviour of chat users", project registration number 33/0012 from 3/03/2025 (2023.04/0012). Also, we would like to thank the reviewers for their precise and concise recommendations that improved the presentation of the results obtained.

References

[1] SeekOut. *AI Recruiting Software Platform*. URL**:** https://www.seekout.com/platform/recruit
[2] Tokar.ua. (2023). *14 facts about Djinni.co: what is the best Ukrainian service for finding a job in IT*. URL**:** https://tokar.ua/read/15964/14-faktiv-pro-djinni-co-chym-zhyve-naykrashchyy-ukr/
[3] Bose, B. (2022). *NLP Text Encoding: A Beginner's Guide*. Medium. URL**:** https://bishalbose294.medium.com/nlp-text-encoding-a-beginners-guide-fa332d715854
[4] DataCamp. *How Transformers Work*. URL**:** https://www.datacamp.com/tutorial/how-transformers-work
[5] Nag, R. (2023). *A Comprehensive Guide to Siamese Neural Networks*. Medium. URL**:** https://medium.com/@rinkinag24/a-comprehensive-guide-to-siamese-neural-networks-3358658c0513
[6] Kaggle. *Resume Dataset*. URL**:** https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset
[7] Kaggle. *Job Vacancy Tweets*. URL**:** https://www.kaggle.com/datasets/prasad22/job-vacancy-tweets
[8] HuggingFace. *Resume-Job Description Fit*. URL**:** https://huggingface.co/datasets/cnamuangtoun/resume-job-description-fit/viewer/default/train
[9] Lytvyn, V., Vysotska, V., Pukach, P., Bobyk, I., & Pakholok, B. (2016). A method for constructing recruitment rules based on the analysis of a specialist's competences. East European Journal of Advanced Technology, (6 (2)), 4-14.
[10] Rzheuskyi, A., Kutyuk, O., Vysotska, V., Burov, Y., Lytvyn, V., & Chyrun, L. (2019, September). The architecture of distant competencies analyzing system for it recruitment. In 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 3, pp. 254-261). IEEE.
[11] Rzheuskyi, A., Kutyuk, O., Voloshyn, O., Kowalska-Styczen, A., Voloshyn, V., Chyrun, L., ... & Rak, T. (2019, September). The intellectual system development of distant competencies analyzing for IT recruitment. In Conference on Computer Science and Information Technologies (pp. 696-720). Cham: Springer International Publishing.
[12] Lytvyn, V., Vysotska, V., & Rzheuskyi, A. (2019). Technology for the psychological portraits formation of social networks users for the IT specialists recruitment based on big five, NLP and big data analysis. In CEUR Workshop Proceedings (pp. 147-171).
[13] Łępicki, M., Latkowski, T., Antoniuk, I., Bukowski, M., Świderski, B., Baranik, G., ... & Kurek, J. (2025). Comparative Evaluation of Sequential Neural Network (GRU, LSTM, Transformer) Within Siamese Networks for Enhanced Job–Candidate Matching in Applied Recruitment Systems. Applied Sciences, 15(11), 5988.
[14] Lytvyn, V., Pukach, P., Bobyk, I., & Vysotska, V. (2016). The method of formation of the status of personality understanding based on the content analysis. East European Journal of Advanced Technology, (5 (2)), 4-12.
[15] Oniani, D., Chandrasekar, P., Sivarajkumar, S., & Wang, Y. (2023). Few-Shot learning for clinical natural language processing using siamese neural networks: algorithm development and validation study. JMIR AI, 2, e44293.
[16] Tian, X., Pavur, R., Han, H., & Zhang, L. (2023). A machine learning-based human resources recruitment system for business process management: using LSA, BERT and SVM. Business Process Management Journal, 29(1), 202-222.
[17] Chen, L. (2024). Research on Improving Recruitment Using Natural Language Processing and AI Technology. https://webofproceedings.org/proceedings_series/ECOM/ICEMEET%202024/ET51.pdf
[18] Deshmukh, A., & Raut, A. (2024). Enhanced Resume Screening for Smart Hiring Using Sentence-Bidirectional Encoder Representations from Transformers (S-BERT). International Journal of Advanced Computer Science & Applications, 15(8).
[19] Kinger, S., Kinger, D., Thakkar, S., & Bhake, D. (2024). Towards smarter hiring: resume parsing and ranking with YOLOv5 and DistilBERT. Multimedia Tools and Applications, 83(35), 82069-82087.
[20] Bevara, R. V. K., Mannuru, N. R., Karedla, S. P., Lund, B., Xiao, T., Pasem, H., ... & Rupeshkumar, S. (2025). Resume2Vec: Transforming Applicant Tracking Systems with Intelligent Resume Embeddings for Precise Candidate Matching. Electronics, 14(4), 794.
[21] Bouhoun, Z., Guerrois, T., Li, X., Baker, M., Elhadji Ille Gado, N., Roumili, E., ... & Plana, R. (2023, June). Information retrieval using domain adapted language models: application to resume documents for HR recruitment assistance. In International Conference on Computational Science and Its Applications (pp. 440-457). Cham: Springer Nature Switzerland.
[22] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.
[23] M. Lan, C. L. Tan, J. Su, and Y. Lu, “Supervised and traditional term weighting methods for automatic text categorization,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 4, pp. 721–735, 2008.
[24] F. Ren and M. G. Sohrab, “Class-indexing-based term weighting for automatic text classification,” Information Sciences, vol. 236, pp. 109–125, 2013.
[25] C. D. Manning, P. Raghavan, and H. Schütze, “Boolean retrieval,” in Introduction to Information Retrieval, Cambridge, UK: Cambridge University Press, 2008, pp. 1–18.
[26] H. Schütze, C. D. Manning, and P. Raghavan, Introduction to Information Retrieval, vol. 39, Cambridge, UK: Cambridge University Press, 2008, pp. 234–265.
[27] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni, “An introduction to information retrieval,” in Web Information Retrieval, Berlin, Germany: Springer, 2013, pp. 3–11.
[28] S. Robertson and H. Zaragoza, “The probabilistic relevance framework: BM25 and beyond,” Foundations and Trends® in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009.

[29] 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, vol. 1 (Long and Short Papers), Minneapolis, MN, USA, June 2019, pp. 4171–4186.
[30] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," arXiv preprint, arXiv:1908.10084, 2019.
[31] M. Łępicki, T. Latkowski, I. Antoniuk, M. Bukowski, B. Świderski, G. Baranik, and J. Kurek, "Comparative evaluation of sequential neural network (GRU, LSTM, Transformer) within Siamese networks for enhanced job–candidate matching in applied recruitment systems," Applied Sciences, vol. 15, no. 11, pp. 5988–5995, 2025.
[32] X. Xue, J. Wang, B. Ma, J. Ren, W. Zhang, S. Gao, and H. Wang, "Fine-grained semantics-enhanced graph neural network model for person-job fit," Entropy, vol. 27, no. 7, pp. 703–710, 2025.
[33] R. Alonso, D. Dessí, A. Meloni, and D. R. Recupero, "A novel approach for job matching and skill recommendation using transformers and the O*NET database," Big Data Research, vol. 36, pp. 100509–100515, 2025.
[34] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in Proceedings of the 4th ACM Conference on Recommender Systems, Barcelona, Spain, September 2010, pp. 39–46.
[35] H. Wang, N. Wang, and D. Y. Yeung, "Collaborative deep learning for recommender systems," in Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, August 2015, pp. 1235–1244.
[36] D. Zhang, J. Liu, H. Zhu, Y. Liu, L. Wang, P. Wang, and H. Xiong, "Job2Vec: Job title benchmarking with collective multi-view representation learning," in Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, November 2019, pp. 2763–2771.
[37] H. Liu and Y. Ge, "Job and employee embeddings: A joint deep learning approach," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 7, pp. 7056–7067, 2022.
[38] A. Hidri, R. Mkhinini Gahar, and M. Sassi Hidri, "What factors distinguish overlapping data job postings? Towards ML-based models for job category factors prediction," Intelligent Decision Technologies, vol. 18, no. 3, pp. 2161–2176, 2024.
[39] M. Yamashita, Y. Li, T. Tran, Y. Zhang, and D. Lee, "Looking further into the future: Career pathway prediction," in Proceedings of the International Conference on Web Search and Data Mining (WSDM) Computational Jobs Marketplace, Virtual, March 2022, pp. 1–8.
[40] J. Zhu, "Learning recruitment-related representations from graphs and sequential data," Ph.D. dissertation, Université Paris-Saclay, Paris, France, 2023. [Online]. Available: https://theses.hal.science/tel-04102832/
[41] M. Yamashita, J. T. Shen, T. Tran, H. Ekhtiari, and D. Lee, "James: Normalizing job titles with multi-aspect graph embeddings and reasoning," in 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA), Thessaloniki, Greece, October 2023, pp. 1–10.
[42] B. Sobol, M. Tonneau, S. Fraiberger, D. Lee, and N. Grinberg, "280 characters to employment: Using Twitter to quantify job vacancies," in Proceedings of the International AAAI Conference on Web and Social Media, vol. 18, Buffalo, NY, USA, May 2024, pp. 1477–1489.
[43] S. Gandhi, R. Nagesh, and S. Das, "Learning skills adjacency representations for optimized reskilling recommendations," in 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, December 2022, pp. 2253–2258.
[44] M. T. Cerilla, A. Santillan, C. J. Vinas, and M. B. D. Fuente, "Career path modeling and recommendations with LinkedIn career data and predicted salary estimations," in Proceedings of the International Conference on Learning Representations (ICLR) Workshop, Virtual, 2023, pp. 1–8. [Online]. Available: https://openreview.net/forum?id=R5NNAThG0i
[45] X. Wang, Z. Jiang, and L. Peng, "A deep-learning-inspired person–job matching model based on sentence vectors and subject–term graphs," Complexity, vol. 2021, pp. 6206288–6206295, 2021.
[46] M. E. Kanakis, R. Khalili, and L. Wang, "Machine learning for computer systems and networking: A survey," ACM Computing Surveys, vol. 55, no. 4, pp. 1–36, 2022.
[47] K. Rekanar, M. J. Hayes, and C. Eising, "Mimicking human attention in driving scenarios for enhanced visual question answering: Insights from eye-tracking and the human attention filter," SSRN Electronic Journal, 2024. [Online]. Available: https://ssrn.com/abstract=5015496
[48] Q. Ni, "Deep neural network model construction for digital human resource management with person–job matching," Computational Intelligence and Neuroscience, vol. 2022, pp. 1418020–1418027, 2022.
[49] Y. Deng, H. Lei, X. Li, and Y. Lin, "An improved deep neural network model for job matching," in 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, May 2018, pp. 106–112.
[50] Z. Wang, W. Wei, C. Xu, J. Xu, and X. L. Mao, "Person-job fit estimation from candidate profile and related recruitment history with co-attention neural networks," Neurocomputing, vol. 501, pp. 14–24, 2022.
[51] A. Thun, "Matching job applicants to free text job ads using semantic networks and natural language inference," M.S. thesis, Uppsala University, Uppsala, Sweden, 2020. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1467916
[52] J. F. B. C. Rojas, "Networks of relations in placement programs serving the transition from school-to-work: The case of secondary school electronics majors in Venezuela," Ph.D. dissertation, New York University, New York, NY, USA, 2001.
[53] H. Chen, C. Mason, Q. Wang, and Y. Zhao, "DBSSM: Deep BERT-based semantic skill matching from resumes to a public skill taxonomy," in Australasian Joint Conference on Artificial Intelligence, Brisbane, Australia, November 2024, pp. 316–328.
[54] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, and H. Xiong, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, June 2018, pp. 25–34.
[55] L. M. Pombo, "Landing on the right job: A machine learning approach to match candidates with jobs applying semantic embeddings," M.S. thesis, Universidade NOVA de Lisboa, Lisbon, Portugal, 2019.
[56] Danylo Levkivskyi, Victoria Vysotska, Lyubomyr Chyrun, Yuriy Ushenko, Dmytro Uhryn, Cennuo Hu, "Agile Methodology of Information Engineering for Semantic Annotations Categorization and Creation in Scientific Articles Based on NLP and Machine Learning Methods", International Journal of Information Engineering and Electronic Business, Vol.17, No.2, pp. 1-50, 2025.

Authors' Profiles

Diana Vyshotravka is a final year bachelor's student at Lviv Polytechnic National University, Department of Information Systems and Networks. She is interested in data analysis and building machine and deep learning models to solve various problems.

Victoria Vysotska is a Professor at the Information Systems and Networks Department of Lviv Polytechnic National University. She defended her Doctoral degree in Technical Science, speciality in «structural, applied and mathematical linguistics», in 2023 on the topic "Analysis and synthesis of computational linguistic systems for processing Ukrainian textual content" Also, she received a PhD degree in Information Technologies from Lviv Polytechnic National University in 2014. She has currently published more than 600 publications. Her main research interests are focused on the identification of disinformation/fakes/propaganda, detection of sources of disinformation and inauthentic behaviour (bots) of coordinated groups based on artificial intelligence, NLP, computer linguistics, data science, system analysis, information technologies, machine learning, cyber security.

Zhengbing Hu: Prof., Deputy Director, International Center of Informatics and Computer Science, Faculty of Applied Mathematics, National Technical University of Ukraine "Kyiv Polytechnic Institute", Ukraine. Adjunct Professor, School of Computer Science, Hubei University of Technology, China. Visiting Prof., DSc Candidate in National Aviation University (Ukraine) from 2019. Primary research interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Dmytro Uhryn graduated from Yuriy Fedkovych Chernivtsi National University, Chernivtsi. Currently, he is a Doctor of Technical Sciences and associate professor at Yuriy Fedkovych Chernivtsi National University. His research interests are data mining, information technologies for decision support, swarm intelligence systems, and industry-specific geographic information systems.

Yuriy Ushenko: World's Top 2% Research Scientist (2020, 2021, 2022, 2024). Nominee of The Photonics100 2025 list by ElectroOptics. Winner of 1000 Talents Plan Program, PhD, DSc, Prof., at Shaoxing University, Shaoxing, Zhejiang Province 312000, China. Professor, Head of Computer Science Department, Chernivtsi National University, Ukraine. His research interests include intelligent information systems design, data mining, pattern recognition and digital image processing, artificial neural networks, laser polarimetry and interferometry.

Kyrylo Smelyakov is currently the Head of the Department of Software Engineering at Kharkiv National University of Radio Electronics (NURE, https://nure.ua/en/staff/kyrylo-smelyakov). In 2012, he completed his doctoral degree in Technical Sciences, specializing in "Mathematical modelling and numerical methods" with a dissertation titled "Models and methods of irregular objects images segmentation for off-line machine vision systems". In 2014, he received a professor's certificate from the Department of Mathematics and Software of ACS at Kharkiv National University of Air Force. Kyrylo has authored over 150 publications, and his primary research interests focus on Information technology, artificial intelligence, machine learning, computer vision, mathematical modelling and numerical methods. His work also encompasses natural language processing (NLP), text processing/recognition, and data science.

How to cite this paper: Diana Vyshotravka, Victoria Vysotska, Zhengbing Hu, Dmytro Uhryn, Yuriy Ushenko, Kyrylo Smelyakov, "Smart Application for Recruiting Based on Natural Language Processing Methods, Transformer Models and Siamese Neural Network Architecture", International Journal of Intelligent Systems and Applications(IJISA), Vol.17, No.5, pp.95-157, 2025. DOI:10.5815/ijisa.2025.05.07