TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC NGÀNH KỸ THUẬT PHẦN MỀM

TÌM HIỂU, ỨNG DỤNG MÔ HÌNH AI PHÁT TRIỂN CÔNG CỤ GỢI Ý VIỆC LÀM THÔNG MINH

GVHD: PGS. TS Trần Đăng Hưng

Sinh viên: Nguyễn Quang Huy

Mã số sinh viên: 2020606068

Hà Nội – 2025

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC NGÀNH KỸ THUẬT PHẦN MỀM

TÌM HIỂU, ỨNG DỤNG MÔ HÌNH AI PHÁT TRIỂN CÔNG CỤ GỢI Ý VIỆC LÀM THÔNG MINH

GVHD: PGS. TS Trần Đăng Hưng

Sinh viên: Nguyễn Quang Huy

Mã số sinh viên: 2020606068

Hà Nội – 2025

MỤC LỤC

LỜI CẢM ƠN iv
DANH MỤC CÁC THUẬT NGỮ, KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT ... v
DANH MỤC BẢNG BIỂU vi
MỞ ĐẦU 1
CHƯƠNG 1: TỔNG QUAN 2
1.1. Giới thiệu chung về trí tuệ nhân tạo (AI) 2
1.1.1 Khái niệm và lịch sử phát triển 2
1.1.2 Các lĩnh vực chính của Trí tuệ nhân tạo 3
1.1.3 Machine Learning và Deep Learning 3
1.2. Ứng dụng AI trong thực tiễn 4
1.2.1 Hệ thống đề xuất (Recommendation Systems) 4
1.2.2 Chatbot và trợ lý ảo 5
1.2.3 Phân tích dữ liệu và dự đoán xu hướng 5
1.3. Thực trạng tìm kiếm việc làm và các nền tảng tìm kiếm việc làm 6
1.3.1 Thách thức trong tìm kiếm việc làm hiện nay 6
1.3.2 Phân tích các nền tảng tìm kiếm việc làm hiện có 6
1.4. Giải pháp giải quyết một số vấn đề trong tìm kiếm việc làm 7
1.4.1 Vai trò của AI trong việc cải thiện quá trình tìm kiếm việc làm 7
1.4.2 Phân tích một số giải pháp hiện có 7
1.4.3 Những thách thức và cơ hội 8
1.5. Đề xuất mô hình và định hướng nghiên cứu 9
1.5.1 Mục tiêu nghiên cứu 9
1.5.2 Phạm vi và giới hạn đề tài 10
1.5.3 Phương pháp nghiên cứu 10
1.6. Kết luận chương 1 11
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 12
2.1. Xử lý ngôn ngữ tự nhiên (NLP) 12

2.1.1 Khái niệm Xử lý Ngôn ngữ Tự nhiên (NLP) 12
2.1.2 Vai trò và tầm quan trọng của NLP 12
2.1.3 Các bước xử lý văn bản cơ bản trong NLP 12
2.2. Term Frequency-Inverse Document Frequency (TF-IDF) 13
2.2.1 Khái niệm 13
2.2.2 Nguyên lý hoạt động 13
2.2.3 Ưu điểm và nhược điểm 15
2.3. Thuật toán K-Nearest Neighbors (KNN) 16
2.3.1 Khái niệm 16
2.3.2 Nguyên lý hoạt động 16
2.3.3 Ưu và nhược điểm của KNN 18
2.4. Deep Learning trong xử lý ngôn ngữ tự nhiên 19
2.4.1 Mô hình BERT (Bidirectional Encoder Representations from Transformers) 19
2.4.2 Ứng dụng của BERT trong các bài toán NLP 22
2.4.3 Hạn chế và hướng phát triển tương lai của BERT 23
2.4.4 So sánh phương pháp truyền thống và Deep Learning trong NLP 23
CHƯƠNG 3: XÂY DỰNG VÀ ĐÁNH GIÁ 25
3.1. Thu thập dữ liệu 25
3.1.1 Nguồn dữ liệu và phương pháp thu thập 25
3.1.2 Các bước tiền xử lý dữ liệu 26
3.2. Xây dựng mô hình TF-IDF kết hợp KNN 28
3.2.1 Thiết kế hệ thống 28
3.2.2 Xây dựng ma trận TF-IDF 29
3.2.3 Áp dụng thuật toán KNN 30
3.2.4 Kết quả thực nghiệm 33
3.3. Xây dựng mô hình xử lý embedding với BERT 35
3.3.1 Thiết kế hệ thống 35

3.3.2 Sử dụng mô hình BERT pre-trained 37
3.3.3 Kết quả thực nghiệm 40
3.4. So sánh và đánh giá hai phương pháp 41
3.4.1 Các metric đánh giá 41
3.4.2 Độ chính xác trong matching 42
3.4.3 Thời gian xử lý 42
3.4.4 Khả năng mở rộng 42
3.4.5 Phân tích ưu nhược điểm mỗi phương pháp 42
3.5. Một số hình ảnh minh hoạ công cụ 43
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 46
TÀI LIỆU THAM KHẢO 48

LỜI CẢM ƠN

Sau 4 năm học tập và rèn luyện tại Trường Đại học Công Nghiệp Hà Nội, đồ án tốt nghiệp này là một dấu ấn quan trọng đánh dấu việc em - một sinh viên đã hoàn thành nhiệm vụ của mình trên ghế giảng đường Đại học. Đầu tiên, với tình cảm sâu sắc, chân thành, cho phép em được bày tỏ lòng biết ơn sâu sắc đến Gia đình, các anh chị, bạn bè đã luôn động viên, khích lệ, tạo điều kiện giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài.

Em cũng xin gửi tới các thầy các cô khoa Công Nghệ Thông Tin Trường Đại học Công Nghiệp Hà Nội lời chào trân trọng, lời chúc sức khỏe và lời cảm ơn sâu sắc. Với sự quan tâm, dạy dỗ, chỉ bảo tận tình chu đáo của thầy cô, đến nay em đã có thể hoàn thành đồ án tốt nghiệp.

Đặc biệt em xin gửi lời cảm ơn chân thành đến thầy PGS. TS. Trần Đăng Hưng và đã tận tình giúp và hướng dẫn em hoàn thành đề tài đồ án. **"Tìm hiểu, ứng dụng mô hình AI phát triển công cụ gợi ý việc làm thông minh".**

Đồng thời, em xin bày tỏ lòng biết ơn đến lãnh đạo Trường Đại học Công Nghiệp Hà Nội đã tạo điều kiện cho em được học tập tại nơi mà em yêu thích, cho em bước vào đời sống thực tế và áp dụng những kiến thức em đã học tại trường và môi trường làm việc mới của em. Qua quá trình học tập em đã tích lũy được rất nhiều kiến thức để chuẩn bị cho công việc sau này cũng như để phát triển thêm bản thân.

Trong quá trình hoàn thành đồ án tốt nghiệp không thể tránh khỏi thiếu sót, kính mong có sự góp ý từ thầy cô.

Em xin chân thành cảm ơn!

DANH MỤC CÁC THUẬT NGỮ, KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

STT	Thuật ngữ, ký hiệu và các chữ viết tắt	Chữ viết đầy đủ
1	AI	Artificial Intelligence
2	KNN	k-Nearest Neigbors
3	TF-IDF	Term Frequency-Inverse Document Frequency
4	BERT	Bidirectional Encoder Representations from Transformers

DANH MỤC BẢNG BIỂU

Hình 2.1: Biểu đồ N-Gram biểu diễn tần suất xuất hiện của các từ liên tiếp trong văn bản 15
Hình 2.2: Ví dụ mình hoạ: hệ trục toạ độ biểu diễn 18
Hình 2.3: Toàn bộ tiến trình pre-training và fine-tuning của BERT 21
Hình 3.1: Dữ liệu thô được thu tập qua Kaggle 25
Hình 3.2: Hàm tiền xử lý văn bản 26
Hình 3.3: Xử lý embedding văn bản để phục vụ import 27
Hình 3.4: Hàm import dữ liệu vào database 27
Hình 3.5: Dữ liệu sau khi import vào database 28
Hình 3.6: Sơ đồ luồng xử lý dữ liệu 29
Hình 3.7: Hàm tính toán TF-IDF cho tập dữ liệu 30
Hình 3.8: Hàm tính khoảng cách cosine 30
Hình 3.9: Hàm tìm K kết quả gần nhất 31
Hình 3.10: Hàm tính TF-IDF cho resume 31
Hình 3.11: Hàm build model TF-IDF 32
Hình 3.12: Hàm sắp xếp và trả về kết quả 32
Hình 3.13: Sơ đồ biểu diễn xử lý dữ liệu 36
Hình 3.14: Sơ đồ xử lí tiền dữ liệu với model BERT pre-trained 37
Hình 3.15: Thiết lập API find_matching_jobs 38
Hình 3.16: Hàm Embedding văn bản 38
Hình 3.17: Hàm tính toán độ tương đồng giữa 2 mảng vector 39
Hình 3.18: Xử lý tính toán mức độ phù hợp giữa resume và từng job 39
Hình 3.19: Trả ra kết quả có độ tương đồng cosine lớn nhất 39
Hình 3.20: Màn danh sách công việc 43
Hình 3.21: Form upload CV 44
Hình 3.22: Kết quả gợi ý việc làm bằng công cụ 44
Hình 3.23: Kết quả công việc phù hợp theo profile ứng viên 45

MỞ ĐẦU

Bài toán gợi ý việc làm phù hợp đang ngày càng trở nên cấp thiết trong bối cảnh thị trường lao động phát triển nhanh chóng và đa dạng hóa ngành nghề. Hệ thống truyền thống thường yêu cầu người tìm việc phải tự mình sàng lọc thông tin, dẫn đến mất nhiều thời gian và công sức. Trong khi đó, ứng dụng trí tuệ nhân tạo (AI) với khả năng phân tích ngữ nghĩa văn bản và tìm kiếm tương đồng hứa hẹn mang đến giải pháp tự động hóa, nâng cao hiệu quả kết nối giữa ứng viên và nhà tuyển dụng.

Việc lựa chọn đề tài "Tìm hiểu, ứng dụng mô hình AI phát triển công cụ gợi ý việc làm thông minh" xuất phát từ nhu cầu thực tiễn kết nối nguồn nhân lực chất lượng cao với cơ hội việc làm phù hợp. Thị trường lao động Việt Nam ngày càng cạnh tranh, đồng thời người tìm việc thường gặp khó khăn trong việc tìm kiếm công việc phù hợp với kỹ năng và kinh nghiệm của bản thân. Bằng cách áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên, các kỹ thuật, thuật toán học máy, đề tài hướng tới xây dựng công cụ hỗ trợ tự động gợi ý việc làm dựa trên phân tích nội dung hồ sơ và mô tả công việc.

Về mặt khoa học, đề tài này mở ra cơ hội áp dụng công nghệ mới vào thị trường tìm kiếm việc làm và tạo nền tảng kiến thức cho nghiên cứu tương tự trong tương lai. Việc xây dựng công cụ gợi ý việc làm thực tiễn không chỉ mang lại nhiều lợi ích cho doanh nghiệp có nhu cầu tuyển dụng mà còn hỗ trợ các ứng viên tìm việc nhanh chóng, phù hợp với năng lực và chuyên môn. Đồng thời, giúp cho mạng lưới kết nối việc làm trở nên hiệu quả và sôi động hơn.

CHƯƠNG 1: TỔNG QUAN

1.1. Giới thiệu chung về trí tuệ nhân tạo (AI)

1.1.1 Khái niệm và lịch sử phát triển

Trí tuệ nhân tạo (Artificial Intelligence - AI) là một lĩnh vực của khoa học máy tính tập trung vào việc xây dựng các hệ thống có khả năng thực hiện các nhiệm vụ đòi hỏi trí thông minh của con người. Những nhiệm vụ này bao gồm việc học hỏi (learning), suy luận (reasoning), giải quyết vấn đề (problem solving), hiểu ngôn ngữ tự nhiên (natural language understanding) và nhận diện thị giác (computer vision).

AI không chỉ đơn thuần là việc lập trình cứng các quy tắc để máy tính thực thi, mà còn liên quan đến việc giúp máy tính có khả năng tự học từ dữ liệu, thích nghi với môi trường mới và đưa ra các quyết định hợp lý trong nhiều tình huống khác nhau.

Khái niệm về AI đã xuất hiện từ những năm 1950 khi nhà khoa học máy tính Alan Turing đặt ra câu hỏi nổi tiếng “Máy móc có thể suy nghĩ được không?” và giới thiệu bài kiểm tra Turing nhằm đánh giá khả năng thông minh của máy móc. Năm 1956, hội thảo Dartmouth do John McCarthy tổ chức đã chính thức đánh dấu sự ra đời của lĩnh vực trí tuệ nhân tạo.

Giai đoạn từ thập niên 1950 đến 1970 chứng kiến những bước phát triển ban đầu của AI với các hệ chuyên gia (expert systems) và các chương trình chơi cờ đơn giản. Tuy nhiên, hạn chế về dữ liệu và sức mạnh tính toán khiến AI bước vào giai đoạn “mùa đông AI” (AI winter) khi kỳ vọng vượt xa khả năng thực tế.

Đến những năm 1990, với sự phát triển mạnh mẽ của phần cứng và các thuật toán học máy (Machine Learning), AI bắt đầu khởi sắc trở lại. Một cột mốc quan trọng là chiến thắng của Deep Blue (IBM) trước nhà vô địch cờ vua Garry Kasparov năm 1997.

Từ năm 2010 trở đi, sự bùng nổ của dữ liệu lớn (Big Data), các bộ xử lý đồ họa (GPU) mạnh mẽ và các mô hình học sâu (Deep Learning) đã đưa AI lên một tầm cao mới. Những thành tựu như AlphaGo đánh bại kỳ thủ cờ vây Lee Sedol năm 2016 hay các mô hình ngôn ngữ lớn như GPT, BERT thể hiện khả năng hiểu và sinh ngôn ngữ tự nhiên đã chứng minh sức mạnh vượt trội của AI hiện đại.

1.1.2 Các lĩnh vực chính của Trí tuệ nhân tạo

Ngày này Trí tuệ nhân tạo (AI) được tìm hiểu, nghiên cứu, triển khải sâu rộng và đa dạng các lĩnh vực, các ngành nghề:

- Học máy (Machine Learning - ML): Tập trung vào phát triển thuật toán cho phép máy học từ dữ liệu mà không cần lập trình tường minh.
- Học sâu (Deep Learning - DL): Nhánh con của ML sử dụng mạng nơ-ron nhiều lớp để tự động trích xuất đặc trưng.
- Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP): Nghiên cứu tương tác giữa máy tính và ngôn ngữ con người.
- Thị giác máy tính (Computer Vision - CV): Chuyển đổi và phân tích hình ảnh, video.
- Hệ thống chuyên gia và quy tắc (Expert Systems & Rule-based): Hệ thống đưa ra quyết định dựa trên tập luật được định nghĩa.
- Robotics: Ứng dụng AI vào điều khiển robot thực thi nhiệm vụ trong môi trường thực.

1.1.3 Machine Learning và Deep Learning

Học máy (machine learning) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Các thuật toán học máy xây dựng một mô hình dựa trên dữ liệu mẫu, được gọi là dữ liệu huấn luyện, để đưa ra dự đoán hoặc quyết định mà không cần được lập trình chi tiết về việc đưa ra dự đoán hoặc quyết định này.

Deep Learning là một nhánh của Machine Learning, trong đó sử dụng kiến trúc mạng nơ-ron nhiều lớp (deep neural networks) để học các biểu diễn phức tạp từ dữ liệu thô. Ưu điểm của Deep Learning so với Machine Learning truyền thống:

- Tự động học đặc trưng: mạng nơ-ron nhiều lớp có thể tự động trích xuất đặc trưng từ dữ liệu, giảm công việc tiền xử lý.
- Hiệu suất cao trên dữ liệu lớn: DL đạt kết quả vượt trội trong các bài toán nhận dạng hình ảnh, âm thanh, ngôn ngữ khi có nhiều dữ liệu và tài nguyên tính toán.
- Khả năng mở rộng: dễ dàng điều chỉnh độ sâu, độ rộng của mạng để phù hợp với yêu cầu bài toán. Nhược điểm:
- Yêu cầu nhiều dữ liệu và tính toán.
- Khó giải thích (black-box), đòi hỏi kỹ thuật giải thích mô hình (model interpretability).

1.2. Ứng dụng AI trong thực tiễn

1.2.1 Hệ thống đề xuất (Recommendation Systems)

Hệ thống đề xuất là một trong những ứng dụng phổ biến nhất của AI trong các nền tảng trực tuyến. AI thu thập dữ liệu hành vi người dùng (lịch sử truy cập, tương tác, đánh giá) và sử dụng các thuật toán học máy để đưa ra đề xuất cá nhân hóa. Ví dụ:

- Netflix và Spotify sử dụng Collaborative Filtering để đề xuất phim, nhạc dựa trên sở thích của người dùng tương tự.
- Amazon áp dụng hybrid recommendation (kết hợp collaborative và content-based) để giới thiệu sản phẩm phù hợp với hồ sơ mua sắm. Với bài toán gợi ý việc làm, hệ thống đề xuất có thể học từ lịch sử ứng tuyển, lượt xem JD và phản hồi của ứng viên để ưu tiên các vị trí phù hợp.

1.2.2 Chatbot và trợ lý ảo

Trong đời sống hàng ngày, các trợ lý ảo như Siri, Alexa, Google Assistant sử dụng AI để hiểu và thực hiện các lệnh từ người dùng, giúp công việc trở nên thuận tiện hơn. Ngoài ra, AI còn xuất hiện trong các hệ thống nhận diện khuôn mặt, lọc spam email, dịch tự động và thậm chí là trong các trò chơi điện tử.Các chatbot AI sử dụng NLP để hiểu và phản hồi câu hỏi của người dùng:

- HR chatbot trên các trang tuyển dụng giúp giải đáp tự động thắc mắc về quy trình ứng tuyển, điều kiện công việc.
- Trợ lý ảo như Google Assistant, Siri hỗ trợ người dùng tìm kiếm thông tin việc làm nhanh chóng qua giọng nói. Công nghệ NLP, đặc biệt là mô hình Transformer, giúp chatbot xử lý ngôn ngữ tự nhiên ngày càng linh hoạt, chính xác hơn.
- ChatGPT: được coi là một bước tiến lớn của AI nói chung và chatbot nói riêng. Một trợ lý đắc lực, có thể tư vấn, hỗ trợ thực hiện các thao tác, công việc nhiều lĩnh vực dựa trên ngữ cảnh prompt người dung nhập vào

1.2.3 Phân tích dữ liệu và dự đoán xu hướng

AI được sử dụng để phân tích lượng lớn dữ liệu tuyển dụng và hồ sơ ứng viên, từ đó rút ra các xu hướng thị trường lao động:

- Phân tích nhu cầu kỹ năng: xác định kỹ năng nào đang có nhu cầu cao qua text mining trên mô tả công việc.
- Dự báo lương và xu hướng tuyển dụng qua time series forecasting.
- Giảm thiểu bias trong tuyển dụng bằng phân tích công bằng (fairness) và explainability. Trong đề tài này, các kỹ thuật phân tích văn bản như TF-IDF và embedding từ mô hình BERT sẽ giúp trích xuất thông tin quan trọng từ resume và JD, hỗ trợ bước phân tích và so khớp.

1.3. Thực trạng tìm kiếm việc làm và các nền tảng tìm kiếm việc làm

1.3.1 Thách thức trong tìm kiếm việc làm hiện nay

Thị trường việc làm ngày càng cạnh tranh với số lượng ứng viên tăng cao và yêu cầu ngày càng khắt khe từ nhà tuyển dụng. Các thách thức chính bao gồm:

Khối lượng thông tin lớn: Người tìm việc thường phải xử lý hàng trăm tin tuyển dụng mỗi ngày, gây mệt mỏi thông tin và làm giảm hiệu quả tìm kiếm. Đa số nền tảng vẫn dựa vào từ khóa để sàng lọc, dễ bỏ sót ứng viên phù hợp nếu hồ sơ không khớp chính xác với từ khóa.

Chất lượng tin không đồng đều: Một số tin đăng thiếu mô tả chi tiết hoặc không rõ ràng yêu cầu công việc.

Thiếu cá nhân hóa: Nền tảng hiện tại thường hiển thị cơ hội chung chung, không phân tích sâu nguyện vọng, kỹ năng và kinh nghiệm thực tế của mỗi ứng viên để đưa ra đề xuất phù hợp.

Rào cản ngôn ngữ và thuật ngữ chuyên môn: Mô tả công việc đa dạng về cách diễn đạt và thuật ngữ, khó so khớp chính xác với hồ sơ ứng viên.

1.3.2 Phân tích các nền tảng tìm kiếm việc làm hiện có

LinkedIn: Mạng xã hội nghề nghiệp lớn nhất với tính năng gợi ý việc làm dựa trên hồ sơ và mạng lưới kết nối. Hạn chế: ưu tiên việc làm theo khu vực và ngành nghề chính, ít hỗ trợ ngôn ngữ tiếng Việt.

Indeed: Tập trung vào tìm kiếm toàn cầu, cho phép lọc việc làm theo từ khóa, vị trí, mức lương. Hạn chế: giao diện đơn giản, thiếu cá nhân hóa sâu.

VietnamWorks: Nền tảng việc làm hàng đầu Việt Nam, chuyên ngành và có hỗ trợ tiếng Việt. Hạn chế: lượng tin cập nhật chưa đa dạng so với nhu cầu thị trường.

TopCV, JobStreet, Glassdoor: Cung cấp đánh giá công ty, mức lương và trải nghiệm tuyển dụng. Hạn chế: dữ liệu đánh giá chủ yếu đến từ các thị trường lớn, ít phản ánh thực tế Việt Nam.

1.4. Giải pháp giải quyết một số vấn đề trong tìm kiếm việc làm

Ứng dụng AI trong gợi ý việc làm

Hệ thống gợi ý việc làm dựa trên AI sử dụng các kỹ thuật học máy, xử lý ngôn ngữ tự nhiên (NLP) và học sâu (Deep Learning) để phân tích hồ sơ ứng viên, lịch sử tìm việc và dữ liệu tuyển dụng. AI có thể rút trích kỹ năng chính, đánh giá độ phù hợp và đề xuất các cơ hội phù hợp theo thang điểm cá nhân hóa

1.4.1 Vai trò của AI trong việc cải thiện quá trình tìm kiếm việc làm

AI đóng vai trò then chốt trong việc nâng cao trải nghiệm người dùng và hiệu quả kết nối ứng viên – nhà tuyển dụng.

AI không chỉ tự động hóa việc sàng lọc hồ sơ mà còn hỗ trợ tối ưu hóa quy trình phỏng vấn, dự đoán xu hướng tuyển dụng và phân tích dữ liệu thị trường lao động thời gian thực. AI giúp giảm thiểu thiên vị con người thông qua các thuật toán khách quan, miễn là dữ liệu huấn luyện được kiểm soát và cân bằng

Bằng việc phân tích ngôn ngữ giữa hồ sơ ứng viên và số lượng công việc lớn, AI làm tốt hơn trong việc cá nhân hoá đề xuất, học thói quen và phản hồi của ứng viên để cải thiện đề xuất theo thời gian. Hệ thống còn chuẩn hóa ngôn ngữ và thuật ngữ chuyên môn, giúp so khớp chính xác hơn giữa ứng viên và yêu cầu công việc.

1.4.2 Phân tích một số giải pháp hiện có

LinkedIn Recruiter: Sử dụng Graph-based Algorithms và ML để gợi ý ứng viên dựa trên mạng lưới kết nối, kinh nghiệm và kỹ năng định danh. Hỗ trợ tính năng “People You May Know” cho tìm ứng viên tiềm năng.

HireVue: Áp dụng AI-driven video interview analysis, phân tích ngôn ngữ và biểu cảm để đánh giá ứng viên. Kết hợp với hệ thống matching dựa trên Keywords và embedding.

Textkernel: Cung cấp công cụ parsing và matching resume – JD với khả năng học từ dữ liệu lịch sử tuyển dụng của doanh nghiệp. Hỗ trợ nhiều ngôn ngữ.

1.4.3 Những thách thức và cơ hội

Việc ứng dụng AI vào các nền tảng tìm kiếm việc làm trong tương lai đặt ra thách thức lớn đầu tiên về thiên vị thuật toán. Các mô hình AI được huấn luyện trên dữ liệu lịch sử từ quá khứ có thể vô tình duy trì hoặc khuếch đại định kiến về giới tính, độ tuổi, chủng tộc hoặc địa lý khi gợi ý công việc. Trong thực tế, nếu dữ liệu tuyển dụng trước đây ưu tiên nhóm ứng viên có một số đặc điểm nhất định, AI sẽ tiếp tục đề xuất việc làm cho nhóm này, dẫn đến sự thiếu đa dạng và cơ hội không công bằng cho các đối tượng khác. Để khắc phục, các nền tảng phải triển khai cơ chế đánh giá và điều chỉnh dữ liệu huấn luyện, đồng thời tự động kiểm tra kết quả đầu ra để phát hiện và giảm thiểu thiên vị.

Bảo mật và quyền riêng tư của người dùng cũng là một rào cản quan trọng. Khi các nền tảng thu thập và phân tích thông tin cá nhân, từ hồ sơ nghề nghiệp cho đến các tương tác trên hệ thống, nguy cơ rò rỉ dữ liệu hoặc lạm dụng thông tin nhằm mục đích thương mại hoặc phân biệt đối xử sẽ gia tăng. Các nhà phát triển cần xây dựng các chính sách bảo vệ dữ liệu nghiêm ngặt, mã hóa đầu cuối và tuân thủ các tiêu chuẩn như GDPR, đồng thời minh bạch với người dùng về cách thu thập và sử dụng thông tin cá nhân.

Một vấn đề khác mà các nền tảng tương lai phải đối mặt là khả năng giải thích quyết định (explainability). Các mô hình học sâu phức tạp có thể đưa ra kết quả gợi ý việc làm chính xác nhưng khó giải thích cơ chế ra quyết định, khiến ứng viên và nhà tuyển dụng khó tin tưởng. Để tạo độ tin cậy, các

hệ thống AI sẽ cần tích hợp các thành phần giải thích như biểu đồ hiệu năng kỹ năng, tóm tắt yếu tố trọng số ảnh hưởng đến lựa chọn công việc và cung cấp dashboard trực quan cho người dùng.

Bên cạnh những thách thức, việc triển khai AI cũng mở ra những cơ hội đáng kể cho các nền tảng tìm việc trực tuyến. Khả năng cá nhân hóa sâu sắc là ưu điểm nổi bật, khi AI phân tích không chỉ kỹ năng chuyên môn, kinh nghiệm và nguyện vọng nghề nghiệp mà còn xét đến xu hướng phát triển ngành, độ phù hợp về văn hóa doanh nghiệp và lộ trình nghề nghiệp dài hạn. Điều này giúp ứng viên nhận được gợi ý phù hợp hơn, đồng thời giảm áp lực lựa chọn giữa hàng loạt tin tuyển dụng không liên quan.

Về phía nhà tuyển dụng, AI cho phép tự động hóa quy trình sàng lọc hồ sơ, rút ngắn thời gian tìm kiếm ứng viên phù hợp và giảm chi phí nhân sự. Hệ thống còn có thể dự đoán xu hướng nhu cầu tuyển dụng dựa trên phân tích dữ liệu thị trường thời gian thực, giúp doanh nghiệp lập kế hoạch tuyển dụng chủ động hơn. Ngoài ra, các nền tảng tương lai có thể cung cấp phân tích chiến lược theo ngành nghề và khu vực, hỗ trợ cả người tìm việc và nhà tuyển dụng nắm bắt cơ hội và điều chỉnh chiến lược phát triển.

Cuối cùng, khi AI được tích hợp chặt chẽ với công nghệ di động và nền tảng mạng xã hội, các hệ thống tìm việc có thể tiếp cận người dùng một cách linh hoạt hơn, từ thông báo gợi ý việc làm theo thời gian thực đến hỗ trợ hướng nghiệp thông qua chatbots thông minh. Sự kết hợp này sẽ mang đến trải nghiệm liền mạch, hiệu quả và phù hợp với xu hướng làm việc từ xa và mô hình hybrid đang ngày càng phổ biến

1.5. Đề xuất mô hình và định hướng nghiên cứu

1.5.1 Mục tiêu nghiên cứu

Một là, phân tích, đánh giá và lựa chọn các phương pháp, các kỹ thuật học máy phù hợp để biểu diễn và so sánh nội dung hồ sơ xin việc (resume) và mô tả công việc (job description).

Hai là, xây dựng và triển khai hai hướng tiếp cận với hai phương pháp: sử dụng kỹ thuật TF-IDF kết hợp thuật toánKNN và sử dụng word embedding từ mô hình BERT pretrained.

Ba là thiết kế công cụ demo với API backend (Flask + MongoDB) và giao diện frontend (Next.js) cho phép tải lên hồ sơ và nhận kết quả gợi ý.

1.5.2 Phạm vi và giới hạn đề tài

Xuyên suốt quá trình tìm hiểu, thực hiện đề tài, toàn bộ nguồn dữ liệu: tập hợp mẫu resume và job description thu thập từ Kaggle và các trang tuyển dụng công khai; quy mô dữ liệu giới hạn trong khoảng 1000 bản ghi.

Tập trung tìm hiểu, nghiên cứu chủ yếu hai phương pháp: kỹ thuật TF-IDF + thuật toán KNN và embedding BERT. Đây là 2 phương pháp tiếp cận được coi là phù hợp trong các công trình xử lý ngôn ngữ tự nhiên, hoàn toàn phù hợp với bài toán gợi ý việc làm

Trên phương diện ngôn ngữ nghiên cứu; công cụ thực hiện và đánh giá kết quả trên dữ liệu tiếng Anh. Các ngôn ngữ khác như Tiếng Việt được xem như hướng đi tiếp theo của đề tài trong tương lai.

Công cụ được thực hiện để minh họa ở mức prototype, không tối ưu cho môi trường sản xuất quy mô lớn.

1.5.3 Phương pháp nghiên cứu

Về tài liệu: thực hiện tìm hiểu các công trình, các nguồn liên quan đến hệ thống gợi ý và xử lý ngôn ngữ tự nhiên trong học máy

Thiết kế và triển khai kỹ thuật TF-IDF, thuật toán KNN dựa trên khoảng cách giữa 2 vectơ số học. Tích hợp mô hình học sâu pretrained BERT để trích xuất embedding trong python nhằm nâng cao tính chính xác dựa trên ngữ nghĩa.

Thử nghiệm và đánh giá kết quả thí nghiệm trên tập dữ liệu ban đầu, thu thập chỉ số Precision, Recall, F1 và so sánh mực độ hiệu quả, tính chính xác của hai phương pháp.

Xây dựng công cụ demo bằng việc phát triển API backend với Flask, lưu trữ dữ liệu và embedding trên MongoDB; triển khai giao diện đơn giản với Next.js để người dùng tương tác và ví dụ trực quan.

1.6. Kết luận chương 1

Trong Chương 1, chúng ta đã khảo sát thực trạng và các công cụ, nền tảng tìm việc làm. Từ đó đưa ra giải pháp sử dụng các thuật toán học máy, mô hình AI để hỗ trợ tìm kiếm việc làm cho ứng viên

Định hướng chương 2: Chương tiếp theo sẽ trình bày chi tiết cơ sở lý thuyết của các thành phần trên:

- Xử lý ngôn ngữ tự nhiên NLP
- Kỹ thuật TF-IDF
- Thuật toán KNN
- Deep learning và mô hình BERT trong xử lý ngôn ngữ tự nhiên

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Xử lý ngôn ngữ tự nhiên (NLP)

2.1.1 Khái niệm Xử lý Ngôn ngữ Tự nhiên (NLP)

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) là một lĩnh vực của trí tuệ nhân tạo (AI) tập trung vào việc nghiên cứu và phát triển các phương pháp giúp máy tính có thể hiểu, phân tích, xử lý và tạo ra ngôn ngữ của con người một cách tự động. Mục tiêu của NLP là xây dựng các hệ thống có khả năng giao tiếp với con người bằng ngôn ngữ tự nhiên, từ đó thu hẹp khoảng cách giữa con người và máy tính.

NLP kết hợp kiến thức từ nhiều lĩnh vực như ngôn ngữ học, khoa học máy tính và học máy để giải quyết các bài toán liên quan đến ngôn ngữ, chẳng hạn như phân tích cú pháp, hiểu ngữ nghĩa, dịch máy, tổng hợp văn bản và trích xuất thông tin.

2.1.2 Vai trò và tầm quan trọng của NLP

Trong bối cảnh dữ liệu văn bản ngày càng phong phú và đa dạng, NLP đóng vai trò then chốt trong việc khai thác và xử lý thông tin từ ngôn ngữ tự nhiên. Các ứng dụng NLP xuất hiện rộng rãi trong nhiều lĩnh vực như tìm kiếm thông tin, hệ thống trợ lý ảo, phân tích cảm xúc khách hàng, dịch máy tự động, chatbot, kiểm tra chính tả và tóm tắt văn bản.

NLP giúp các tổ chức, doanh nghiệp tiết kiệm thời gian và chi phí khi xử lý khối lượng lớn dữ liệu văn bản phi cấu trúc, đồng thời nâng cao trải nghiệm người dùng thông qua các hệ thống giao tiếp tự nhiên, thông minh hơn.

2.1.3 Các bước xử lý văn bản cơ bản trong NLP

Quy trình xử lý ngôn ngữ tự nhiên thường bao gồm nhiều bước từ tiền xử lý đến phân tích sâu hơn về mặt ngữ nghĩa và ngữ cảnh. Một số bước xử lý văn bản cơ bản trong NLP gồm:

Tiền xử lý văn bản (Preprocessing): Là bước đầu tiên nhằm chuẩn hóa và làm sạch dữ liệu văn bản. Các kỹ thuật phổ biến bao gồm chuyển chữ hoa thành chữ thường, loại bỏ dấu câu, từ dừng (stopwords), tokenization (tách từ), stemming và lemmatization (chuẩn hóa từ gốc).

Biểu diễn văn bản (Text Representation): Chuyển đổi văn bản từ dạng chuỗi ký tự sang dạng vector số để máy tính có thể xử lý. Các phương pháp truyền thống như Bag of Words, TF-IDF, và các phương pháp hiện đại hơn như word embeddings (Word2Vec, GloVe) và contextual embeddings (BERT, ELMo) được sử dụng rộng rãi.

Phân tích cú pháp và ngữ nghĩa (Syntax and Semantic Analysis): Xác định cấu trúc ngữ pháp của câu (Part-of-Speech tagging, Dependency Parsing) và phân tích ý nghĩa ngữ cảnh của các từ và cụm từ trong câu.

Ứng dụng vào bài toán cụ thể: Sau khi dữ liệu văn bản đã được xử lý và biểu diễn phù hợp, mô hình NLP sẽ được áp dụng vào các bài toán như phân loại văn bản, trích xuất thực thể, phân tích cảm xúc hay trả lời câu hỏi.

2.2. Term Frequency-Inverse Document Frequency (TF-IDF)

2.2.1 Khái niệm

TF-IDF là một kĩ thuật cơ bản và được sử dụng kết hợp với rất nhiều thuật toán khác để xử lý ngôn ngữ: phân tích dữ liệu văn bản, tìm kiếm thông tin, tóm tắt văn bản...

TF-IDF là viết tắt của thuật ngữ tiếng Anh “Term Frequency – Inverse Document Frequency”, TF-IDF là trọng số của một từ trong văn bản thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản.

2.2.2 Nguyên lý hoạt động

TF-IDF thường được sử dụng làm bước đầu trong các bài toán phân loại văn bản, truy vấn tìm kiếm thông tin và trích xuất đặc trưng trong các mô hình máy học. Khi triển khai, cần kết hợp TF-IDF với các bước tiền xử lý như

tokenization, loại bỏ từ dừng và chuẩn hóa từ gốc để nâng cao chất lượng biểu diễn. Ngoài ra, trong các hệ thống truy vấn tìm kiếm, TF-IDF giúp xếp hạng tài liệu theo mức độ liên quan, từ đó cải thiện độ chính xác của kết quả trả về.

TF-IDF đánh giá mức độ quan trọng của một từ trong một tài liệu so với toàn tập văn bản:

TF (Term Frequency): dùng để ước lượng tần xuất xuất hiện của từ trong văn bản. Tuy nhiên với mỗi văn bản thì có độ dài khác nhau, vì thế số lần xuất hiện của từ có thể nhiều hơn. Vì vậy số lần xuất hiện của từ sẽ được chia độ dài của văn bản (tổng số từ trong văn bản đó)

$$TF(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

IDF- Inverse Document Frequency: dùng để ước lượng mức độ quan trọng của từ đó như thế nào. Khi tính tần số xuất hiện tf thì các từ đều được coi là quan trọng như nhau. Tuy nhiên có một số từ thường được được sử dụng nhiều nhưng không quan trọng để thể hiện ý nghĩa của đoạn văn, ví dụ :

- Từ nối: và, nhưng, tuy nhiên, vì thế, vì vậy, …
- Giới từ: ở, trong, trên, …
- Từ chỉ định: ấy, đó, nhỉ, …

Vì vậy ta cần giảm đi mức độ quan trọng của những từ đó bằng cách sử dụng IDF:

$$IDF(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

Cơ số logarit trong công thức này không thay đổi giá trị của 1 từ mà chỉ thu hẹp khoảng giá trị của từ đó. Vì thay đổi cơ số sẽ dẫn đến việc giá trị của các

từ thay đổi bởi một số nhất định và tỷ lệ giữa các trọng lượng với nhau sẽ không thay đổi. (nói cách khác, thay đổi cơ số sẽ không ảnh hưởng đến tỷ lệ giữa các giá trị IDF). Tuy nhiên việc thay đổi khoảng giá trị sẽ giúp tỷ lệ giữa IDF và TF tương đồng để dùng cho công thức TF-IDF như bên dưới.

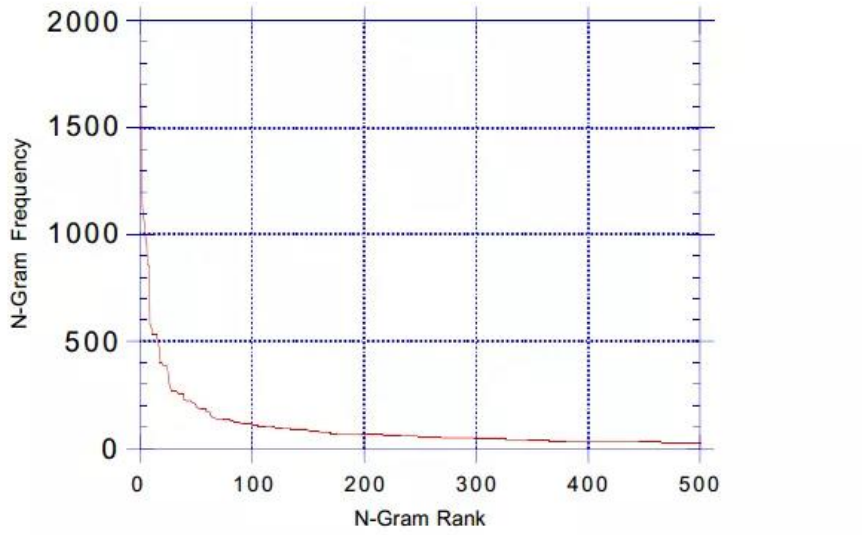
Giá trị TF-IDF:

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó).

2.2.3 Ưu điểm và nhược điểm

Trong quá trình phân tích ngôn ngữ, tài liệu, văn bản; các từ quan trọng luôn đi kèm thường xuyên với các từ khác.



Hình 2.1: Biểu đồ N-Gram biểu diễn tần suất xuất hiện của các từ liên tiếp trong văn bản

Có nghĩa là luôn có một tập các từ mà tần số xuất hiện, sử dụng nhiều hơn các từ khác, điều này đúng trong bất kì ngôn ngữ nào. Chính vì vậy ta cần có một phương pháp để làm mịn đường cong tần số trên hay là việc cân bằng mức độ quan trọng giữa các từ.

Với việc tận dụng tính toán mức độ xuất hiện của từ ngữ trong văn bản, kỹ thuật này đem đến sự đơn giản trong áp dụng thực tiễn, chú trọng vào các từ ít xuất hiện nhưng mang một tính đặc trưng nhất định.

Mặc dù TF-IDF có nhiều ưu điểm, phương pháp này cũng tồn tại một số hạn chế. Đầu tiên, TF-IDF không xem xét ngữ cảnh và thứ tự của từ trong câu, dẫn đến việc bỏ qua thông tin ngữ nghĩa quan trọng. Tiếp theo, kết quả TF-IDF phản ánh tần suất mà không thể phân biệt giữa các dạng khác nhau của cùng một từ (ví dụ: "học" và "học tập"), trừ khi thực hiện bước xử lý từ gốc (stemming/lemmatization). Cuối cùng, đối với tập tài liệu rất lớn hoặc đa ngôn ngữ, việc tính toán IDF có thể trở nên tốn kém về bộ nhớ và thời gian tính toán.

2.3. Thuật toán K-Nearest Neighbors (KNN)

2.3.1 Khái niệm

Thuật toán K-Nearest Neighbors (KNN) là một phương pháp học máy có giám sát không tham số, được sử dụng phổ biến cho cả bài toán phân loại và hồi quy. KNN không xây dựng mô hình tham số mà thực hiện dựa trên việc ghi nhớ toàn bộ tập dữ liệu huấn luyện, sau đó đưa ra dự đoán cho mẫu mới dựa vào khoảng cách đến các điểm gần nhất trong tập huấn luyện. Đây là một thuật toán thân thiện với người mới học bởi tính trực quan và đơn giản trong cách thức hoạt động.

Trong ngữ cảnh phân loại, KNN xác định nhãn của một điểm dữ liệu mới bằng cách xem xét các nhãn của K điểm lân cận gần nhất trong không gian đặc trưng, sau đó chọn nhãn có số phiếu bầu nhiều nhất. Đối với bài toán hồi quy, giá trị dự đoán được tính trung bình hoặc trọng số của các giá trị của K láng giềng này.

2.3.2 Nguyên lý hoạt động

Nguyên lý cơ bản của KNN dựa trên giả thiết: các mẫu dữ liệu có tính chất tương tự sẽ nằm gần nhau trong không gian đặc trưng. Khi một điểm dữ

liệu mới xuất hiện, thuật toán sẽ tính toán khoảng cách giữa điểm này và tất cả các điểm trong tập huấn luyện, thường sử dụng khoảng cách Euclid hoặc Manhattan. Sau đó, thuật toán sắp xếp các điểm theo thứ tự tăng dần của khoảng cách và chọn K điểm đầu tiên làm láng giềng gần nhất.

Bước quan trọng tiếp theo là xác định giá trị của K. Một K quá nhỏ có thể gây nhạy cảm với nhiễu trong dữ liệu, trong khi K quá lớn có thể làm mất đi chi tiết phân chia giữa các lớp. Giá trị K thường được chọn thông qua kỹ thuật kiểm định chéo (cross-validation) hoặc các phương pháp tối ưu hóa siêu tham số khác.

Đối với bài toán phân loại, nhãn của mẫu mới được xác định bằng nhãn chiếm đa số trong K láng giềng, trong khi với bài toán hồi quy, giá trị dự đoán thường là trung bình của các giá trị của K láng giềng. Một số biến thể còn sử dụng trọng số tỷ lệ nghịch với khoảng cách để các láng giềng gần hơn có ảnh hưởng lớn hơn.

Các độ đo khoảng cách (Euclidean, Cosine, Manhattan)

- **Euclidean Distance** (khoảng cách Euclid): độ dài trực tiếp giữa hai vector trong không gian nhiều chiều.
- **Cosine Similarity**: đo góc giữa hai vector, cho biết mức độ tương đồng hướng dù bỏ qua độ dài.
- **Manhattan Distance** (khoảng cách L1): tổng giá trị tuyệt đối hiệu các thành phần, phù hợp với dữ liệu sparse.

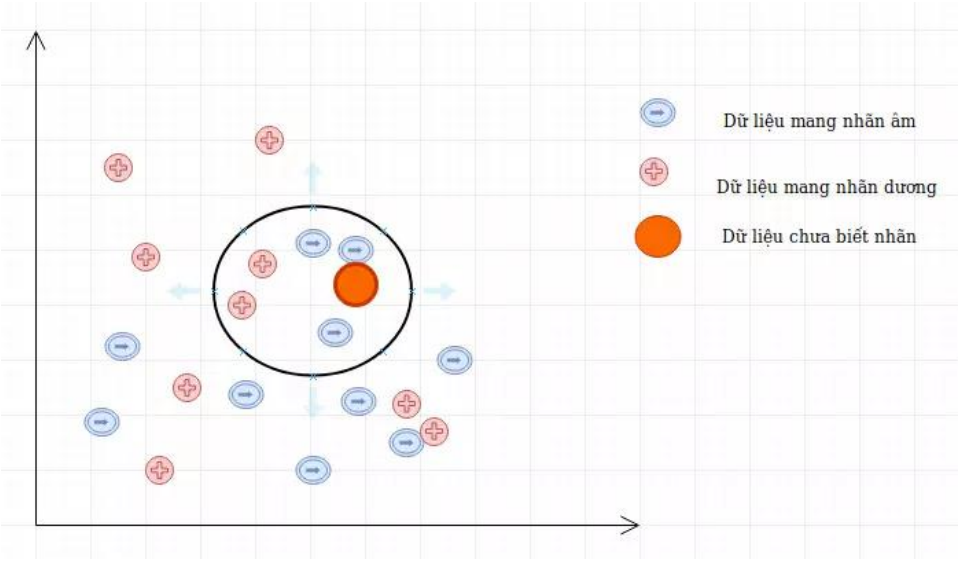
Ví dụ:

Giả sử ta có D là tập các dữ liệu đã được phân loại thành 2 nhãn (+) và (-) được biểu diễn trên trục tọa độ và một điểm dữ liệu mới A chưa biết nhãn. Vậy làm cách nào để chúng ta có thể xác định được nhãn của A là (+) hay (-)

Có thể thấy cách đơn giản nhất là so sánh tất cả các đặc điểm của dữ liệu A với tất cả tập dữ liệu học đã được gắn nhãn và xem nó giống cái nào nhất, nếu dữ liệu (đặc điểm) của A giống với dữ liệu của điểm mang nhãn (+) thì

điểm A mang nhãn (+), nếu dữ liệu A giống với dữ liệu nhãn (-) hơn thì nó mang nhãn (-), trông có vẻ rất đơn giản nhưng đó là những gì mà KNN làm.

Trong trường hợp của KNN, thực tế nó không so sánh dữ liệu mới (không được phân lớp) với tất cả các dữ liệu khác, thực tế nó thực hiện một phép tính toán học để đo khoảng cách giữa dữ liệu mới với tất cả các điểm trong tập dữ liệu học D để thực hiện phân lớp. Phép tính khoảng cách giữa 2 điểm có thể là Euclidian, Manhattan, trọng số, Minkowski, …



Hình 2.2: Ví dụ mình hoạ: hệ trục toạ độ biểu diễn

2.3.3 Ưu và nhược điểm của KNN

Một trong những ưu điểm nổi bật của KNN là tính đơn giản và dễ hiểu, giúp người dùng nhanh chóng nắm bắt và triển khai thuật toán. Thuật toán này không yêu cầu giai đoạn huấn luyện phức tạp, bởi toàn bộ công việc được dời sang pha suy luận (lazy learning), do đó tiết kiệm thời gian huấn luyện ban đầu.

KNN đặc biệt hiệu quả khi áp dụng cho dữ liệu phi tuyến, bởi nó không đưa ra bất kỳ giả thiết phân phối nào cho dữ liệu. Khả năng điều chỉnh linh hoạt bằng cách thay đổi hàm khoảng cách và giá trị K giúp thuật toán dễ dàng thích ứng với nhiều loại bài toán.

Thuật toán KNN đòi hỏi bộ nhớ lưu trữ toàn bộ tập dữ liệu huấn luyện, dẫn đến chi phí lưu trữ và tính toán cao khi dữ liệu lớn. Việc tính khoảng cách cho mọi điểm trong tập huấn luyện cũng gây tốn thời gian, đặc biệt với số lượng mẫu.

KNN nhạy cảm với dữ liệu nhiễu và ngoại lệ; một láng giềng bất thường có thể làm sai lệch kết quả dự đoán, nhất là khi K nhỏ. Ngoài ra, thuật toán chịu ảnh hưởng tiêu cực từ lời nguyền chiều không gian (curse of dimensionality), khi số chiều tăng lên, khoảng cách Euclid mất tính phân biệt, làm giảm hiệu suất phân loại

2.4. Deep Learning trong xử lý ngôn ngữ tự nhiên

Deep Learning, hay còn gọi là học sâu, là một nhánh của trí tuệ nhân tạo (AI) và học máy (Machine Learning) chuyên nghiên cứu các mô hình mạng nơ-ron nhân tạo với nhiều tầng (deep neural networks). Trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP), Deep Learning đã mang lại những đột phá vượt bậc nhờ khả năng tự động học các đặc trưng ngôn ngữ phức tạp mà không cần quá nhiều kỹ thuật thủ công.

Trước khi Deep Learning trở nên phổ biến, các mô hình xử lý ngôn ngữ chủ yếu dựa vào các kỹ thuật truyền thống như thống kê, mô hình n-gram, hay vector hóa từ (TF-IDF, Bag of Words). Tuy nhiên, những phương pháp này gặp nhiều hạn chế trong việc nắm bắt ngữ cảnh dài hạn và quan hệ ngữ nghĩa sâu sắc giữa các từ. Sự xuất hiện của các mô hình mạng nơ-ron hồi tiếp (RNN), mạng nơ-ron tích chập (CNN) và đặc biệt là mô hình Transformer đã giúp NLP tiến một bước dài về hiệu quả và độ chính xác.

2.4.1 Mô hình BERT (Bidirectional Encoder Representations from Transformers)

BERT (Bidirectional Encoder Representations from Transformers) là một trong những mô hình tiêu biểu nhất cho sự thành công của Deep Learning trong NLP, được Google giới thiệu vào năm 2018. Khác với các mô hình

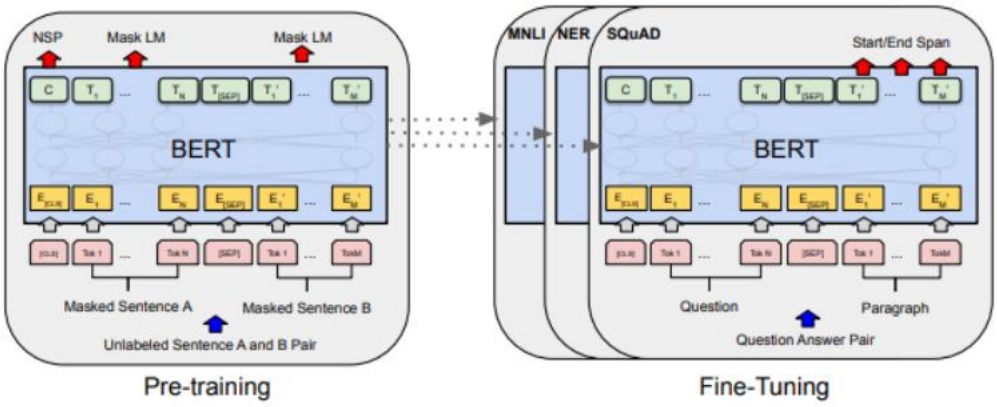
trước đó chỉ xử lý ngữ cảnh theo chiều từ trái sang phải hoặc phải sang trái, BERT sử dụng cơ chế attention hai chiều (bidirectional attention) giúp hiểu ngữ cảnh đầy đủ từ cả hai phía của một từ trong câu.

Cơ chế attention của Transformer sẽ truyền toàn bộ các từ trong câu văn đồng thời vào mô hình một lúc mà không cần quan tâm đến chiều của câu. Do đó Transformer được xem như là huấn luyện hai chiều (bidirectional) mặc dù trên thực tế chính xác hơn chúng ta có thể nói rằng đó là huấn luyện không chiều (non-directional). Đặc điểm này cho phép mô hình học được bối cảnh của từ dựa trên toàn bộ các từ xung quanh nó bao gồm cả từ bên trái và từ bên phải.

Điểm đặc biệt của BERT nằm ở phương pháp huấn luyện "Masked Language Model" (MLM), trong đó một số từ trong câu sẽ bị che đi và mô hình được yêu cầu dự đoán lại những từ bị ẩn đó. Nhờ vậy, BERT có khả năng học được các mối liên hệ ngữ nghĩa phức tạp trong văn bản một cách hiệu quả hơn. Bên cạnh đó, BERT còn được huấn luyện với nhiệm vụ "Next Sentence Prediction" (NSP), giúp mô hình hiểu được quan hệ giữa các câu liên tiếp, điều này rất hữu ích cho các bài toán như trả lời câu hỏi hay phân loại văn bản.

Fine-tuning model BERT

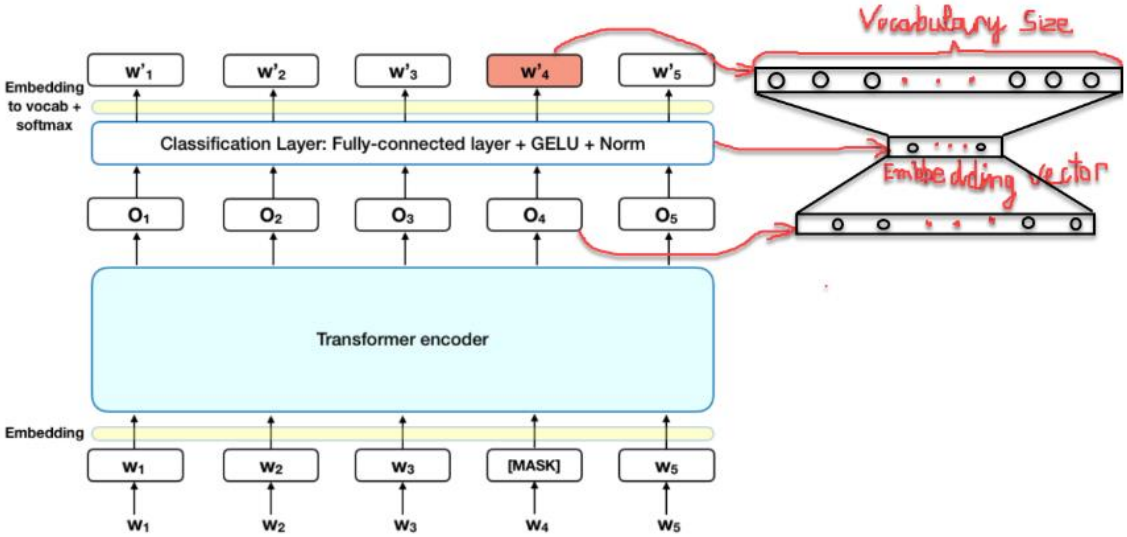
Một điểm đặc biệt ở BERT mà các model embedding trước đây chưa từng có đó là kết quả huấn luyện có thể fine-tuning được. Chúng ta sẽ thêm vào kiến trúc model một output layer để tùy biến theo tác vụ huấn luyện.



Hình 2.3: Toàn bộ tiến trình pre-training và fine-tuning của BERT

Masked ML (MLM)

Masked ML là một tác vụ cho phép chúng ta fine-tuning lại các biểu diễn từ trên các bộ dữ liệu unsupervised-text bất kỳ. Chúng ta có thể áp dụng Masked ML cho những ngôn ngữ khác nhau để tạo ra biểu diễn embedding cho chúng. Các bộ dữ liệu của tiếng anh có kích thước lên tới vài vài trăm tới vài nghìn GB được huấn luyện trên BERT đã tạo ra những kết quả khá ấn tượng



Khoảng 15 % các token của câu input được thay thế bởi [MASK] token trước khi truyền vào model đại diện cho những từ bị che dấu (masked). Mô hình sẽ dựa trên các từ không được che (non-masked) dấu xung quanh [MASK] và đồng thời là bối cảnh của [MASK] để dự báo giá trị gốc

của từ được che dấu. Số lượng từ được che dấu được lựa chọn là một số ít (15%) để tỷ lệ bối cảnh chiếm nhiều hơn (85%).

Bản chất của kiến trúc BERT vẫn là một mô hình seq2seq gồm 2 phase encoder giúp embedding các từ input và decoder giúp tìm ra phân phối xác suất của các từ ở output. Kiến trúc Transfomer encoder được giữ lại trong tác vụ Masked ML. Sau khi thực hiện self-attention và feed forward ta sẽ thu được các véc tơ embedding ở output là O1,O2,…,O5

Để tính toán phân phối xác suất cho từ output, chúng ta thêm một Fully connect layer ngay sau Transformer Encoder. Hàm softmax có tác dụng tính toán phân phối xác suất. Số lượng units của fully connected layer phải bằng với kích thước của từ điển.

Cuối cùng ta thu được véc tơ nhúng của mỗi một từ tại vị trí MASK sẽ là embedding véc tơ giảm chiều của véc tơ Oi sau khi đi qua fully connected layer như mô tả trên hình vẽ bên phải.

2.4.2 Ứng dụng của BERT trong các bài toán NLP

Với khả năng biểu diễn ngữ nghĩa mạnh mẽ, BERT đã được ứng dụng rộng rãi trong nhiều bài toán NLP quan trọng như phân loại văn bản, trích xuất thực thể (NER), phân tích cảm xúc, trả lời câu hỏi (Question Answering) và đặc biệt là các hệ thống tìm kiếm thông tin. Các mô hình tiền huấn luyện dựa trên BERT thường chỉ cần tinh chỉnh thêm với một lượng dữ liệu nhỏ là có thể đạt được kết quả vượt trội trong các nhiệm vụ cụ thể.

Ví dụ, trong bài toán tìm kiếm, BERT giúp cải thiện đáng kể độ chính xác khi hiểu rõ hơn ý định thực sự của truy vấn người dùng thay vì chỉ khớp từ khóa đơn giản. Trong các hệ thống chatbot hay trợ lý ảo, BERT giúp mô hình trả lời tự nhiên và phù hợp hơn với ngữ cảnh, nhờ khả năng nắm bắt quan hệ giữa các câu và cụm từ một cách toàn diện.

2.4.3 Hạn chế và hướng phát triển tương lai của BERT

Mặc dù đạt được nhiều thành tựu, BERT vẫn tồn tại một số hạn chế nhất định. Kích thước mô hình lớn dẫn đến chi phí tính toán cao, yêu cầu tài nguyên phần cứng mạnh mẽ để huấn luyện và triển khai. Ngoài ra, BERT tiêu chuẩn xử lý các đoạn văn bản có độ dài tối đa 512 token, gây khó khăn cho các bài toán cần xử lý văn bản dài.

Để khắc phục những hạn chế này, nhiều biến thể của BERT đã ra đời như DistilBERT, RoBERTa, ALBERT nhằm giảm kích thước mô hình, tăng hiệu suất tính toán mà vẫn giữ được độ chính xác cao. Các hướng nghiên cứu mới cũng tập trung vào việc cải thiện khả năng xử lý ngữ cảnh dài hạn, giảm độ phức tạp mô hình và mở rộng khả năng hiểu biết ngôn ngữ đa ngữ.

2.4.4 So sánh phương pháp truyền thống và Deep Learning trong NLP

Phương pháp truyền thống trong xử lý ngôn ngữ tự nhiên thường dựa trên các kỹ thuật thống kê và mô hình hóa đơn giản. Các phương pháp như Bag of Words hay TF-IDF tập trung biểu diễn văn bản dưới dạng các vector tần suất, nhưng lại bỏ qua thông tin về thứ tự từ và ngữ cảnh. Do đó, chúng chỉ phù hợp cho các tác vụ cơ bản và kém hiệu quả khi xử lý những bài toán yêu cầu hiểu biết sâu sắc về ngữ nghĩa và ngữ cảnh dài hạn.

Ngược lại, Deep Learning cho phép mô hình học tự động các đặc trưng ngôn ngữ thông qua quá trình huấn luyện với dữ liệu lớn. Các mạng nơ-ron hồi tiếp (RNN, LSTM) giúp xử lý chuỗi dữ liệu và giữ lại thông tin từ các từ trước đó, trong khi các mô hình dựa trên Transformer và attention như BERT lại vượt trội nhờ khả năng song song hóa và nắm bắt quan hệ giữa các từ bất kỳ trong câu. Kết quả là, Deep Learning không chỉ cải thiện độ chính xác mà còn mở ra khả năng xử lý các tác vụ phức tạp hơn như dịch máy, tóm tắt văn bản và trả lời câu hỏi tự động.

Tuy nhiên, chi phí tính toán và yêu cầu dữ liệu lớn là thách thức lớn đối với Deep Learning, trong khi các phương pháp truyền thống vẫn giữ ưu thế về

tốc độ xử lý và dễ triển khai cho các bài toán đơn giản. Do đó, việc lựa chọn phương pháp phù hợp cần dựa trên đặc thù bài toán và nguồn lực sẵn có.

Tiêu chí	**Phương pháp truyền thống (TF-IDF, KNN)**	**Phương pháp Deep Learning (BERT, Transformer)**
Khả năng nắm ngữ cảnh	Hạn chế, chỉ dựa trên tần suất/đồng xuất hiện	Cao, nắm bắt mối quan hệ dài hạn và hai chiều
Yêu cầu dữ liệu	Ít, phù hợp với tập nhỏ	Nhiều, cần dữ liệu lớn để pre-train
Độ giải thích	Dễ giải thích trọng số từ	Black-box, cần kỹ thuật XAI
Tốc độ xử lý	Nhanh, tài nguyên thấp	Chậm, tốn tài nguyên
Ứng dụng phù hợp	Prototyping, dữ liệu nhỏ/medium	Ứng dụng sản xuất, độ chính xác cao

Bảng 2.1: So sánh phương pháp truyền thống và Deep Learning trong NLP

CHƯƠNG 3: XÂY DỰNG VÀ ĐÁNH GIÁ

3.1. Thu thập dữ liệu

3.1.1 Nguồn dữ liệu và phương pháp thu thập

Dữ liệu chính gồm hai bộ:

- **Job Descriptions**: thu thập từ Kaggle (ví dụ tập "Resume Dataset"), các trang tuyển dụng công khai (Indeed, LinkedIn) qua scraping hoặc API nếu có.
- **Resumes mẫu**: lấy từ Kaggle (tập "Resume Dataset"), forum, blog chuyên ngành.

Phương pháp thu thập:

- Làm sạch dữ liệu thô bằng loại bỏ HTML, CSS, script.
- Lưu trữ thô dưới dạng JSON/CSV để thuận tiện xử lý.
- Sử dụng Python với Mongodb import trực tiếp dữ liệu từ file csv đã được xử lý
- Đối với phương pháp sử dụng mô hình BERT, do việc xử lý tiêu tốn một lượng lớn tài nguyên cho nên ta tận dụng sức mạnh của google collab thay vì thiết bị vật lý có sẵn

Dữ liệu thô khi thập được từ trên Kaggle là một file csv bao gồm hơn 800 bản ghi là các job description:

company_name	job_description	position_title	description	model_response
Google	minimum qualifications	Sales Specialist	2727	{
Apple	description	Apple Solutions Consult	828	{
Netflix	its an amazing time to be joining netflix as we continue to transform entertainment	Licensing Coordinator -	3205	{
Robert Half	description	Web Designer	2489	{
TrackFive	at trackfive weve got big goals were on a mission to revolutionize recruiting with	Web Developer	3167	{
DesignUps	designups is a nashville based design and interactive agency at designups we have a strong	Frontend Web Develop	892	{
Equisolve, Inc.	about the position	Remote Website Desig	3471	{
Zander Insurance Agency	job description	Web Designer	2896	{
Tuff	tuff is a growth marketing team working with clients to drive growth by creating managing	Web Designer	5143	{
General Dynamics Information Technology	type of requisition regular	SR. Web Designer	4023	{
Sony Music Entertainment	web developer	Web Developer	4181	{
Snapshot Interactive	snapshot is looking for a back end developer to join our organization and make an	Web Developer	3842	{
Deloitte	we are seeking a senior ui designer who relishes in methodically applied color palettes and	Senior UI Designer	4484	{
Themesoft Inc	design develop and test high quality software features by understanding user needs and	Wordpress Web Develc	647	{
Western Governors University	if youre passionate about building a better future for individuals communities and our	UI Web Designer	3733	{
Sedgwick	the apply with seek option will be utilized for international applicants mainly australia if this	Senior Web Designer (R	4027	{
Ohio ENT & Allergy Physicians	description	Chief Executive Officer	3202	{
Confidential Jobs	about the company	Executive Vice Presider	569	{
RiverVista Health and Wellness	chief executive officer rivervista	CEO	2239	{
Salt Creek Capital	salt creek capital is seeking an experienced operating executive to join the firm as an	CEO	1328	{
Ascend Innovations	job description	CEO, Positivly	5403	{
ExecHQ	company overview	Chief Executive Officer	2900	{
Confidential	the role of the chief executive officer ceo is to provide leadership that ensures the	Chief Executive Officer	1973	{
On Time Talent Solutions	chief executive officer healthcare	Chief Executive Officer	2516	{
Vistage Worldwide	vistage ceo coach	CEO Coach	1978	{

Hình 3.1: Dữ liệu thô được thu tập qua Kaggle

Cấu trúc dữ liệu Job Descriptions

- job_id: định danh nhiệm vụ.
- position title: tiêu đề công việc.
- job_info: các thông tin của job
- description: nội dung chi tiết yêu cầu, nhiệm vụ.
- embedding

3.1.2 Các bước tiền xử lý dữ liệu

- **Làm sạch văn bản**: đưa văn bản về chữ thường, loại bỏ ký tự đặc biệt, HTML tags, chuỗi dư thừa.
- **Tokenization**: sử dụng nltk.word_tokenize để tách từ.
- **Loại bỏ Stop Words**: sử dụng bộ từ dừng từ NLTK cho tiếng Anh.
- **Gán nhãn bổ sung**: phân loại từ, câu theo pos_tag, nếu cần, đánh dấu kỹ năng, trình độ bằng NER hoặc rules đơn giản.

```
def preprocess_text_v2(text):
    try:
        text = text.lower()
        text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

        sentences = sent_tokenize(text)

        processed_sentences = []

        for sent in sentences:
            if any(criteria in sent for criteria in ['skills', 'education']):
              words = word_tokenize(sent)
              words = [word for word in words if word not in stop_words]
              tagged_words = pos_tag(words)
              filtered_words = [word for word, tag in tagged_words if tag not in ['DT', 'IN', 'TO', 'PRP', 'WP']]
              processed_sentences.append(" ".join(filtered_words))

        return " ".join(processed_sentences)
    except Exception as e:
        print(f"Error in text preprocessing: {e}")
        return text
```

Hình 3.2: Hàm tiền xử lý văn bản

Embedding hoá văn bản, bổ trợ cho việc tính toán qua model bert:

```python
def get_bert_embedding_batch(texts):
    processed_texts = [preprocess_text_v2(text) for text in texts]
    try:
        max_length = 512
        inputs = tokenizer(processed_texts, return_tensors='pt', truncation=True,
                           max_length=max_length, padding='max_length').to(device)
        with torch.no_grad():
            outputs = model(**inputs)
        embeddings = outputs.last_hidden_state[:, 0, :].cpu().numpy()
        return embeddings.tolist()
    except Exception as e:
        logger.error(f"Error generating BERT embedding batch: {e}")
        return None
```

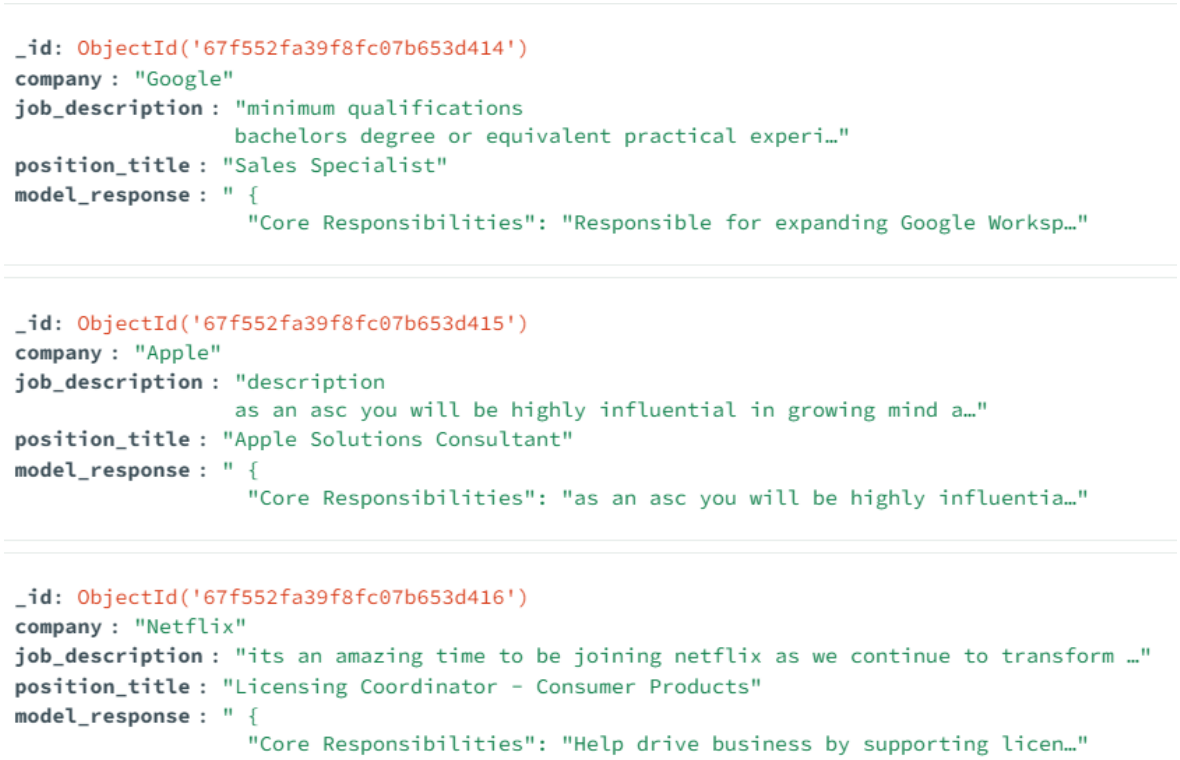
Hình 3.3: Xử lý embedding văn bản để phục vụ import

Thực hiện import văn bản đã qua xử lý và embedding để import vào database:

```python
def import_jobs_batched(csv_file_path, batch_size=32):
    try:
        df = pd.read_csv(csv_file_path)
        total_jobs = len(df)
        processed_count = 0
        jobs_data = []

        for i in range(0, total_jobs, batch_size):
            batch_df = df.iloc[i:i + batch_size]
            batch_descriptions = batch_df.iloc[:, 1].tolist() # Lấy cột 'job_description' (cột thứ 2))

            embeddings_batch = get_bert_embedding_batch(batch_descriptions)

            if embeddings_batch:
                for j, embedding in enumerate(embeddings_batch):
                    row = batch_df.iloc[j]
                    job_entry = {
                        "company": row.iloc[0],
                        "job_description": row.iloc[1],
                        "position_title": row.iloc[2],
                        "model_response": row.iloc[4],
                        "embedding": embedding
                    }
                    jobs_data.append(job_entry)
                    processed_count += 1
                    if processed_count % 100 == 0 or processed_count == total_jobs:
                        logger.info(f"Processed {processed_count}/{total_jobs} jobs")
            else:
                logger.warning(f"Could not generate embeddings for batch starting at index {i}")

        if jobs_data:
            job_collection.insert_many(jobs_data)
            return {
                "success": True,
                "message": f"{len(jobs_data)} jobs imported successfully with BERT embeddings"
            }
        else:
            return {"error": "No valid jobs to import"}

    except Exception as e:
        logger.error(f"Error importing jobs: {e}")
        return {"error": str(e)}
```

Hình 3.4: Hàm import dữ liệu vào database

Bản chất của việc sử dụng google collab đó là dung tài nguyên của google để thực hiện pre-train model, khi đã embedding được dữ liệu ta thực hiện import về database



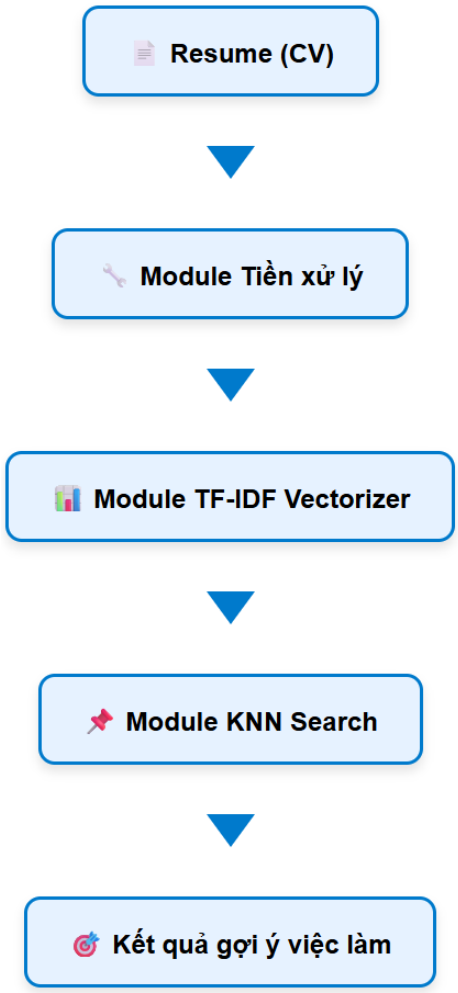
Hình 3.5: Dữ liệu sau khi import vào database

3.2. Xây dựng mô hình TF-IDF kết hợp KNN

Dựa trên cơ sở lý thuyết kỹ thuật TF-IDF và thuật toán KNN thực hiện xử lý dữ liệu, phân tích. Mục tiêu là tìm kiếm được các kết quả job có danh mục sát với resume đầu vào nhất có thể

3.2.1 Thiết kế hệ thống

- Mô hình sơ đồ luồng dữ liệu: resume → tiền xử lý → ma trận TF-IDF → KNN → kết quả gợi ý.
- Thành phần chính: module tiền xử lý, module TF-IDF vectorizer, module KNN search.



Hình 3.6: Sơ đồ luồng xử lý dữ liệu

3.2.2 Xây dựng ma trận TF-IDF

- **Trích xuất đặc trưng từ job descriptions**: gom tất cả JD, tính TF-IDF trên corpus để tạo ma trận (D = số JD, V = số từ vựng).
- **Trích xuất đặc trưng từ resume**: áp dụng vectorizer đã train trên JD lên từng resume đầu vào, thu được vector TF-IDF độ dài V.

```python
def calculate_tfidf_docs(documents):
    idf_values = {}
    total_docs = len(documents)

    for doc in documents:
        unique_words = set(doc)
        for word in unique_words:
            if word not in idf_values:
                idf_values[word] = 0
            idf_values[word] += 1

    for word in idf_values:
        idf_values[word] = math.log(total_docs / idf_values[word])

    tfidf_documents = []
    for doc in documents:
        tf_values = {}
        word_count = Counter(doc)
        doc_length = len(doc)
        for word, count in word_count.items():
            tf_values[word] = count / doc_length

        tfidf_doc = {}
        for word in tf_values:
            tfidf_doc[word] = tf_values[word] * idf_values.get(word, 0)
        tfidf_documents.append(tfidf_doc)

    return tfidf_documents
```

Hình 3.7: Hàm tính toán TF-IDF cho tập dữ liệu

3.2.3 Áp dụng thuật toán KNN

Dùng khoảng cách Cosine để tính toán khoảng cách giữa 2 vectơ. Các số liệu tính toán từ module TF-IDF sẽ được vectơ hoá rồi so sánh

```python
def cosine_distance(vector1, vector2):
    common_keys = set(vector1.keys()) & set(vector2.keys())
    dot_product = sum(vector1[key] * vector2[key] for key in common_keys)
    magnitude1 = math.sqrt(sum(vector1[key]**2 for key in vector1))
    magnitude2 = math.sqrt(sum(vector2[key]**2 for key in vector2))
    if magnitude1 == 0 or magnitude2 == 0:
        return 1  # zero vector
    return 1 - dot_product / (magnitude1 * magnitude2)
```

Hình 3.8: Hàm tính khoảng cách cosine

Lọc kết quả dựa trên k kết quả gần nhất:

```
def find_knn(query_vector, vectors, k):
    distances = []
    for i, vector in enumerate(vectors):
        distance = cosine_distance(query_vector, vector)
        distances.append((i, distance))
    distances.sort(key=lambda x: x[1])
    return distances[:k]
```

Hình 3.9: Hàm tìm K kết quả gần nhất

Chi tiết các bước xử lý dữ liệu:

Bước 1: Thiết lập hàm, input là file resume với định dạng pdf hoặc word.

Bước 2: Tiền xử lý file resume, tính TF-IDF cho file resume.

Với resume, chúng ta có thể sử dụng luôn tập IDF của job description, hoặc có thể chỉ sử dụng TF để tính toán

```
# calc TF-IDF of resume
# TF
tf_values = {}
for word in resume_words:
    if word not in tf_values:
        tf_values[word] = 0
    tf_values[word] += 1

for word in tf_values:
    tf_values[word] /= len(resume_words)

resume_tfidf = {}
for word in tf_values:
    # only use TF if not idf value
    resume_tfidf[word] = tf_values[word]
```

Hình 3.10: Hàm tính TF-IDF cho resume

Bước 3: Dựng model TF-IDF cho tập dữ liệu và lấy dữ liệu của các job trong job collection. Có thể lưu dữ liệu tính sẵn vào pickle, từ lần sau có thể lấy kết quả đó ra để tính toán

```python
def build_tfidf_model():
    all_jobs = list(job_embedding.find({}, {'_id': 1, 'job_description': 1, 'position_title': 1, 'model_response': 1, 'company': 1}))

    if not all_jobs:
        client.close()
        return None, None, []

    job_texts = [preprocess_text(job.get('job_description', '') + job.get('position_title','') + job.get('model_response', '')) for job in all_jobs]
    job_words = [[word for word in text.split()] for text in job_texts]

    tfidf_documents = calculate_tfidf_docs(job_words)

    # client.close()
    return all_jobs, tfidf_documents
```

Hình 3.11: Hàm build model TF-IDF

Bước 4: Vector hoá và so sánh TF-IDF của resume và job collection bằng khoảng cách cosine

Bước 5: Sắp xếp danh sách kết quả lấy ra các cặp vector có khoảng cách cosine nhỏ nhất và trả về kết quả

```python
    # find k nearest neighbors
    neighbors = find_knn(resume_tfidf, tfidf_documents, k)
    # print(neighbors)
    # get matching job best
    matching_jobs = []
    for idx, distance in neighbors:
        if idx < len(all_jobs):
            job = all_jobs[idx]
            job_id = str(job.get('_id', 'unknown'))
            position_title = job.get('position_title')
            company = job.get('company')
            similarity_score = float(1 - distance)
            benefit = None
            model_response_str = job.get('model_response')
            if(model_response_str):
                try:
                    model_response_dict = json.loads(job['model_response'])
                    benefit = model_response_dict.get('Compensation and Benefits')
                except (TypeError, json.JSONDecodeError):
                    benefit = None

            matching_jobs.append({
                'id': job_id,
                'position_title': position_title,
                'company': company,
                'benefit': "Negotiable" if benefit == "N/A" else benefit,
                'similarity_score': similarity_score,
            })

    return matching_jobs
```

Hình 3.12: Hàm sắp xếp và trả về kết quả

3.2.4 Kết quả thực nghiệm

Ví dụ cụ thể với một resume mẫu: resume của ứng viên

***Trích xuất**

- Dữ liệu thô là văn bản được lấy ra từ file PDF.
- Ví dụ, một đoạn văn bản thô: "Motivated, enthusiastic educational leader with 15+ years' experience fostering a cohesive student learning atmosphere conducive to learning. My core qualifications include being a licensed reading specialist in multiple states, strong classroom management, public speaking skills, CPR Certified and F.E.M.A. certified for emergency management situations."

***Tiền xử lý:**

- Bước này làm sạch và chuẩn bị văn bản.
- **Chuyển thành chữ thường:** "motivated, enthusiastic educational leader with 15+ years' experience..."
- **Loại bỏ ký tự đặc biệt:** "motivated enthusiastic educational leader with 15 years experience..." (Lưu ý: "15+" có thể được giữ lại hoặc loại bỏ tùy thuộc vào biểu thức chính quy cụ thể)
- **Tách từ (Tokenization):** ['motivated', 'enthusiastic', 'educational', 'leader', 'with', '15', 'years', 'experience', ...]
- **Loại bỏ stop words (từ dừng):** ['motivated', 'enthusiastic', 'educational', 'leader', '15', 'years', 'experience', ...] (ví dụ, "with" bị loại bỏ)
- **Kết quả của Tiền xử lý:** Một danh sách các từ đã được làm sạch, hoặc một chuỗi văn bản đã được làm sạch sẵn sàng cho việc chuyển đổi thành vector.

***Chuyển đổi TF-IDF:**

- Văn bản đã tiền xử lý được chuyển thành một vector số.
- **Tần suất Từ (TF - Term Frequency):**

- Đếm số lần mỗi từ xuất hiện trong CV.
- Ví dụ:
    - "teacher": 5
    - "student": 7
    - "lesson": 4
    - ...

- Tần suất nghịch đảo của văn bản (IDF - Inverse Document Frequency):

- Để tính IDF, chúng ta cần một tập hợp các văn bản (ví dụ: tập hợp nhiều CV và Mô tả Công việc (JD)). Chúng ta có ~800 văn bản.
- Nếu "teacher" xuất hiện trong 200 văn bản, IDF("teacher") = log(800/200) ≈ 1.61
- Nếu "student" xuất hiện trong 500 văn bản, IDF("student") = log(800/500) ≈ 0.69
- Nếu "lesson" xuất hiện trong 100 văn bản, IDF("lesson") = log(800/100) ≈ 2.30

- **Tính toán TF-IDF:**
    - Nhân giá trị TF với giá trị IDF cho mỗi từ.
    - Ví dụ:
        1. TF-IDF("teacher") = 5 * 1.61 = 8.05
        2. TF-IDF("student") = 7 * 0.69 = 4.83
        3. TF-IDF("lesson") = 4 * 2.30 = 9.20
- **Biểu diễn Vector TF-IDF:**
    - CV bây giờ là một vector, trong đó mỗi chiều tương ứng với một từ trong từ vựng của chúng ta (tất cả các từ duy nhất trong tất cả CV và JD), và giá trị của chiều đó thể hiện tầm quan trọng của từ đó trong CV.
    - Ví dụ (đơn giản hóa, giả sử từ vựng chỉ có "teacher", "student", "lesson"):

***So khớp KNN:**

Giả sử chúng ta có một Mô tả Công việc (JD) và nó cũng được chuyển đổi thành một vector TF-IDF: Vector JD: [2.0, 5.0, 1.0]

- **Độ tương tự Cosin:**

- Chúng ta tính độ tương tự cosin giữa vector CV và vector JD.
- Độ tương tự Cosin = (8.05*2.0 + 4.83*5.0 + 9.20*1.0) / (||[8.05, 4.83, 9.20]|| * ||[2.0, 5.0, 1.0]||) (trong đó || || là độ dài của vector)
- Điều này cho chúng ta một điểm số từ -1 đến 1, trong đó 1 có nghĩa là CV và JD rất giống nhau.

- KNN:

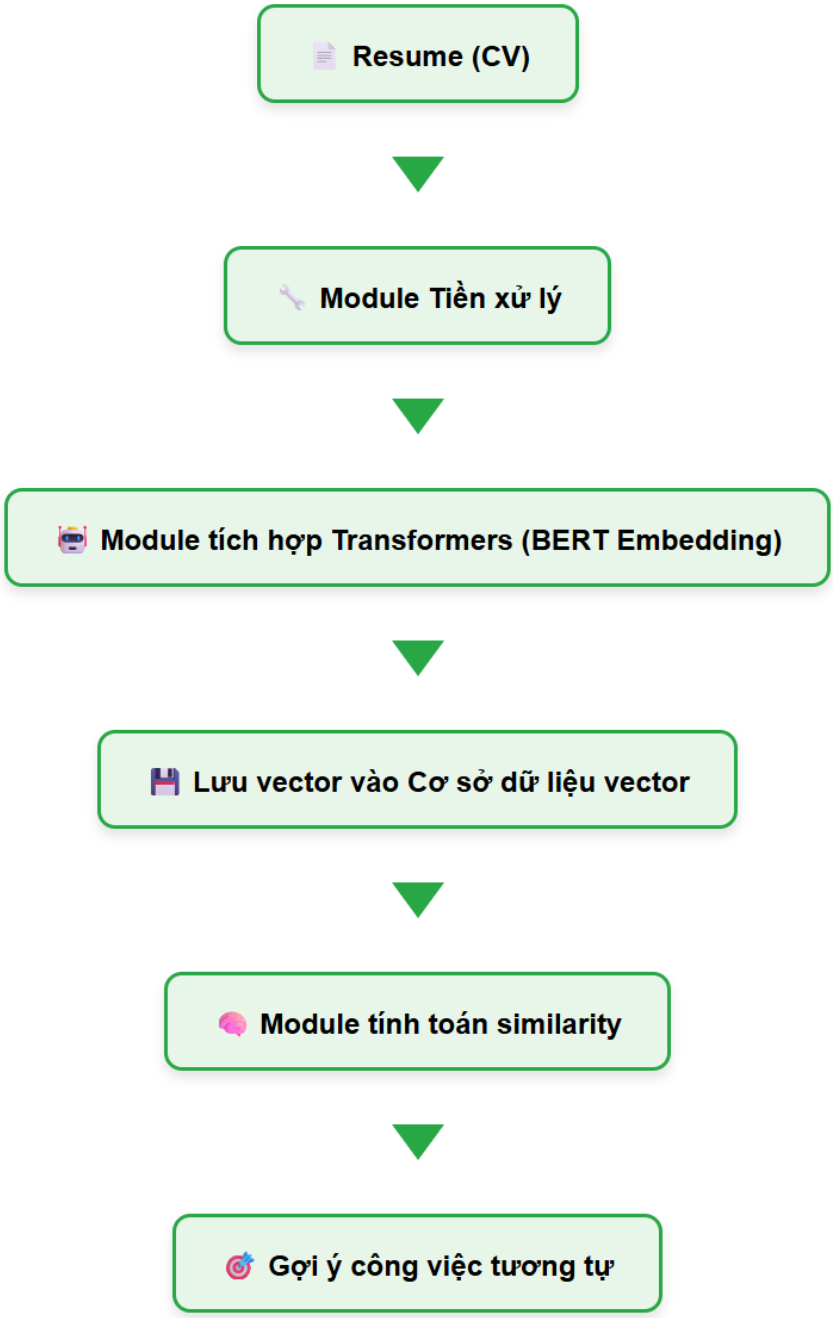
- Nếu chúng ta sử dụng k=5, chúng ta sẽ tính độ tương tự cosin giữa CV và mọi JD trong cơ sở dữ liệu của chúng ta.
- Sau đó, chúng ta chọn 5 JD hàng đầu có điểm số độ tương tự cosin cao nhất.
- 5 JD hàng đầu này là "những người hàng xóm gần nhất" và được đề xuất cho ứng viên.

3.3. Xây dựng mô hình xử lý embedding với BERT

Dựa trên cơ sở lý thuyết tại chương 2, ta áp dụng mô hình BERT để tính toán với đầu vào tương tự phương pháp 1. Do mức độ phức tạp của thuật toán khá cao nên ta sẽ sử dụng model có sẵn từ thư viện transformers. Mục tiêu là sẽ trả ra kết quả có sự phù hợp về ngữ nghĩa tốt hơn so với phương pháp sử dụng TF-IDF + KNN

3.3.1 Thiết kế hệ thống

- Sơ đồ luồng: resume → tiền xử lý → embedding BERT → lưu vector → truy vấn tìm job tương tự.
- Thành phần: Module tích hợp Transformers, module tính toán similarity, cơ sở dữ liệu vector.



Hình 3.13: Sơ đồ biểu diễn xử lý dữ liệu

3.3.2 Sử dụng mô hình BERT pre-trained

Dùng model BERT tiền xử lý dữ liệu data collection và lưu vào database:

Hình 3.14: Sơ đồ xử lí tiền dữ liệu với model BERT pre-trained

Các bước thực hiện xử lý dữ liệu

Bước 1: Thiết lập hàm, input đầu vào là file resume với định dạng pdf hoặc docx,

```python
@app.route('/jobs/match-resume-bert', methods=['POST'])
def find_matching_jobs_bert():
    if 'resume' not in request.files:
        return jsonify({"error": "No CV file uploaded"}), 400

    resume_file = request.files['resume']
    if resume_file.filename == '':
        return jsonify({"error": "No selected CV file"}), 400

    top_n = request.args.get('k', default = 5, type=int)

    resume_path = os.path.join(app.config['UPLOAD_FOLDER'], secure_filename(resume_file.filename))

    try:
        resume_file.save(resume_path)

        # Trích xuất text từ PDF
        resume_text = extract_text_from_file(resume_path)
```

Hình 3.15: Thiết lập API find_matching_jobs

Bước 2: Trích xuất văn bản trong resume, tiền xử lý và embedding hoá văn bản trong resume.

```python
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
def get_bert_embedding(text):
    try:
        # Cắt text nếu quá dài (BERT có giới hạn 512 tokens)
        max_length = 512        You, 2 weeks ago • feat: find with bert

        # Tokenize và encode
        inputs = tokenizer(text, return_tensors='pt', truncation=True, max_length=max_length, padding='max_length')

        # Không tính gradient vì chỉ cần forward pass
        with torch.no_grad():
            outputs = model(**inputs)

        # Lấy vector embedding của token [CLS] (đại diện cho cả câu)
        embeddings = outputs.last_hidden_state[:, 0, :].numpy().flatten()

        return embeddings.tolist()
    except Exception as e:
        # logger.error(f"Error generating BERT embedding: {e}")
        return None
```

Hình 3.16: Hàm Embedding văn bản

Bước 3: Lấy ra toàn bộ các job có trong database rồi so sánh độ tương đồng cosine giữa các vector

Hàm tính toán mức độ phù hợp giữa resume và các job trong database

```python
def cosine_similarity(a, b):
    a = np.array(a)
    b = np.array(b)
    return np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))
```

Hình 3.17: Hàm tính toán độ tương đồng giữa 2 mảng vector

```python
# Lấy tất cả job từ MongoDB
jobs = list(job_embedding.find({}, {'_id': 1, 'company': 1, 'position_title': 1, 'job_description': 1, 'model_response': 1, 'embedding': 1}))

# Tính cosine similarity giữa CV và mỗi job
matched_jobs = []
for job in jobs:
    # Kiểm tra job có embedding không
    if 'embedding' not in job:
        print(job)
    benefit = None
    model_response_str = job.get('model_response')
    if(model_response_str):
        try:
            model_response_dict = json.loads(job['model_response'])
            benefit = model_response_dict.get('Compensation and Benefits')
        except (TypeError, json.JSONDecodeError):
            benefit = None

    similarity = cosine_similarity(cv_embedding, job['embedding'])
    matched_jobs.append({
        'id': str(job['_id']),
        'company': job['company'],
        'position_title': job['position_title'],
        'similarity_score': float(similarity),
        'benefit': "Negotiable" if benefit == 'N/A' else benefit
    })
```

Hình 3.18: Xử lý tính toán mức độ phù hợp giữa resume và từng job

Bước 4: Sắp xếp và lấy ra các job có mức độ tương đồng với resume là lớn nhất

```python
# Sắp xếp theo độ tương đồng giảm dần
matched_jobs.sort(key=lambda x: x['similarity_score'], reverse=True)

os.remove(resume_path)

# Trả về top N kết quả
return jsonify({
    "isSuccess": True,
    "data": matched_jobs[:top_n],
    "errorCode": None,
    "message": None,
})
```

Hình 3.19: Trả ra kết quả có độ tương đồng cosine lớn nhất

3.3.3 Kết quả thực nghiệm

Ví dụ cụ thể với một resume mẫu: resume ứng viên Data Scientist với kỹ năng Python, Machine Learning, SQL.

***Token hóa:**

- Văn bản thô được token hóa bằng BERT tokenizer. Điều này khác với việc tách từ đơn giản; nó chia các từ thành các đơn vị dưới từ.
- Ví dụ:
  - Văn bản thô: "Motivated, enthusiastic educational leader"
  - BERT tokens: ['motivated', ',', 'enthusiastic', 'educational', 'leader']
- BERT tokenizer cũng thêm các token đặc biệt như [CLS] ở đầu và [SEP] ở cuối câu.

***Tạo Embedding:**

- Mô hình BERT lấy các token làm đầu vào và tạo ra một biểu diễn vector dày đặc (embedding) cho toàn bộ CV.
- Hàm get_bert_embedding trong code của bạn trích xuất embedding từ đầu ra của token [CLS].
- Ví dụ: CV Embedding: [0.12, -0.54, 0.88, ..., -0.21] (một vector có số chiều là 768 với model base)
- Tương tự, mỗi JD trong cơ sở dữ liệu của chúng ta cũng sẽ được chuyển đổi thành một BERT embedding.

***Tính toán Độ tương tự:**

- Chúng ta tính độ tương tự cosin giữa embedding CV và embedding của mỗi JD.
- Đây là công thức độ tương tự cosin giống như trong KNN, nhưng bây giờ được áp dụng cho các BERT embedding, nắm bắt thông tin ngữ nghĩa tốt hơn.

***Xếp hạng và Đề xuất:**

- Chúng ta xếp hạng các JD dựa trên điểm số độ tương tự cosin của chúng với CV.
- N JD hàng đầu (ví dụ: 5 JD hàng đầu) có điểm số tương tự cao nhất được đề xuất cho ứng viên.

***Điểm khác biệt chính và Trực quan hóa**

- **TF-IDF:**
  - Vector thưa (nhiều giá trị 0).
  - Mỗi chiều tương ứng với một từ.
  - Nắm bắt tần suất và tầm quan trọng của từ.
  - KNN sử dụng khoảng cách cosin trên các vector thưa này để tìm các tài liệu tương tự.
- **BERT:**
  - Vector dày đặc (hầu hết các giá trị khác 0).
  - Các chiều không có ý nghĩa từ trực tiếp; chúng đại diện cho các mối quan hệ ngữ nghĩa phức tạp.
  - Nắm bắt ý nghĩa theo ngữ cảnh của từ.
  - Độ tương tự cosin trên các BERT embedding dày đặc tìm các tài liệu tương tự về mặt ngữ nghĩa.

3.4. So sánh và đánh giá hai phương pháp

3.4.1 Các metric đánh giá

- **Precision@K, Recall@K, F1-Score**: đo lường khả năng gợi ý chính xác các vị trí phù hợp trong top K kết quả.
- **Relevance**: mức độ liên quan trung bình của các job được gợi ý so với hồ sơ ứng viên.
- **Diversity**: độ đa dạng trong danh sách job, đảm bảo không chỉ tập trung vào một nhóm kỹ năng nhất định.

3.4.2 Độ chính xác trong matching

- So sánh Precision, Recall của hai phương pháp trên tập test.
- Phân tích các ngưỡng tương đồng để tìm trade-off giữa độ chính xác và phủ.

3.4.3 Thời gian xử lý

- Đo thời gian tạo ma trận TF-IDF và truy vấn KNN trung bình cho một resume.
- Đo thời gian inference embedding BERT và tính toán similarity cho một resume.
- So sánh hiệu năng và xác định giới hạn xử lý theo quy mô dữ liệu.

3.4.4 Khả năng mở rộng

- TF-IDF + KNN: chi phí lưu trữ ma trận sparse cao, khó tăng tốc cho bộ dữ liệu lớn.
- BERT: yêu cầu tính toán embedding nặng, nhưng có thể batch và sử dụng GPU/TPU để mở rộng.
- Đề xuất sử dụng Elasticsearch k-NN hoặc FAISS để nâng cao khả năng truy vấn với vector lớn.

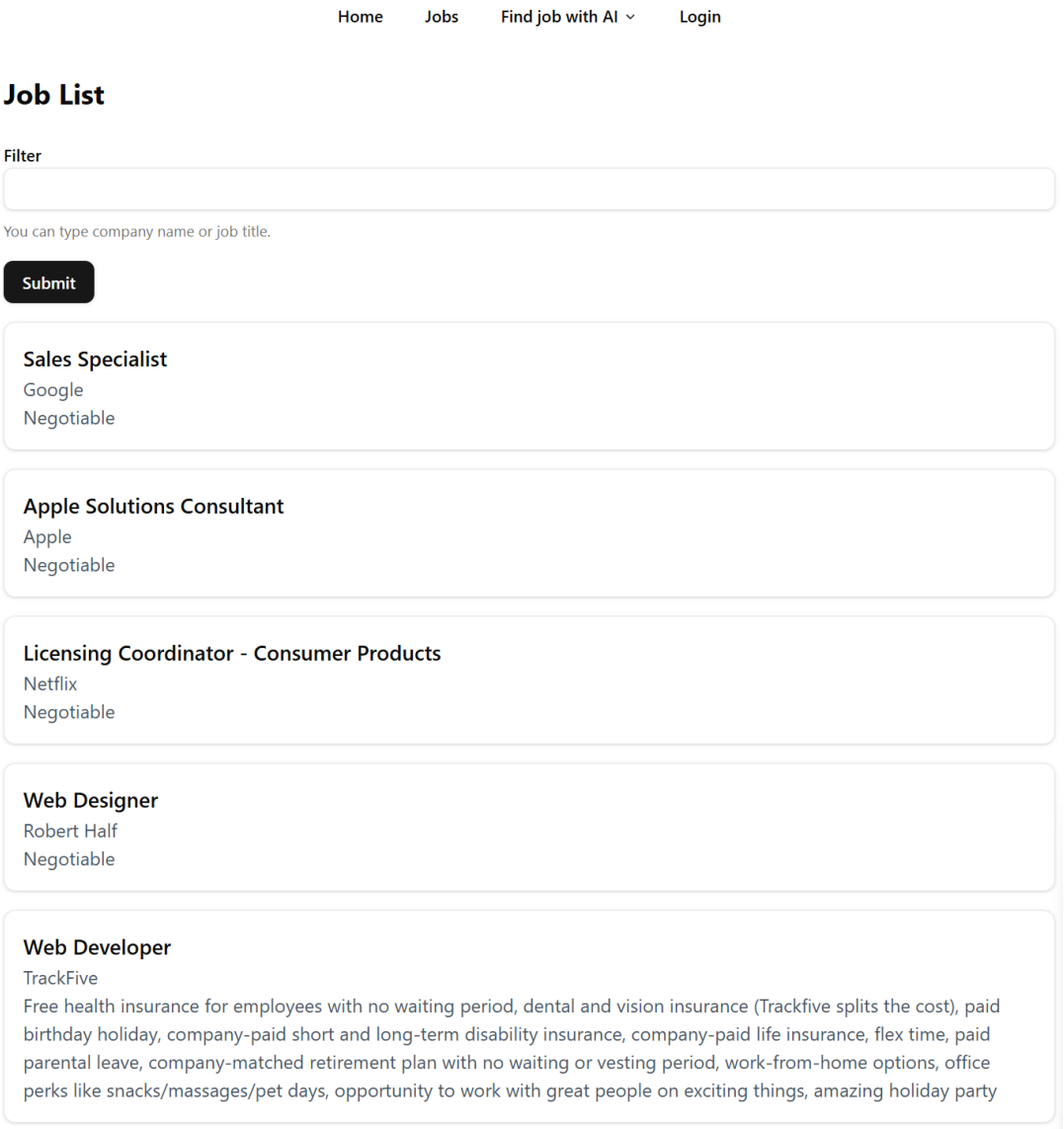
3.4.5 Phân tích ưu nhược điểm mỗi phương pháp

Tiêu chí	TF-IDF + KNN	BERT Embedding
Độ chính xác	Tốt với từ khóa rõ ràng, nhưng thiếu ngữ cảnh	Cao, nắm bắt semantics, phù hợp khi ngữ cảnh quan trọng
Tốc độ xử lý	Nhanh, tài nguyên CPU thấp	Chậm hơn, cần GPU để tối ưu
Phức tạp triển khai	Đơn giản, ít phụ thuộc external	Phức tạp, cần cài đặt thư viện và model pre-trained
Khả năng mở rộng	Khó mở rộng cho bộ dữ liệu rất lớn	Có thể mở rộng nhờ batching và indexing vector
Giải thích kết quả	Rõ ràng dựa trên trọng số từ	Khó giải thích, cần kỹ thuật XAI

Bảng 3.1: Phân tích ưu nhược điểm giữa 2 phương pháp TF-IDF và BERT Embedding

3.5. Một số hình ảnh minh hoạ công cụ

Danh sách công việc trong cơ sở dữ liệu:

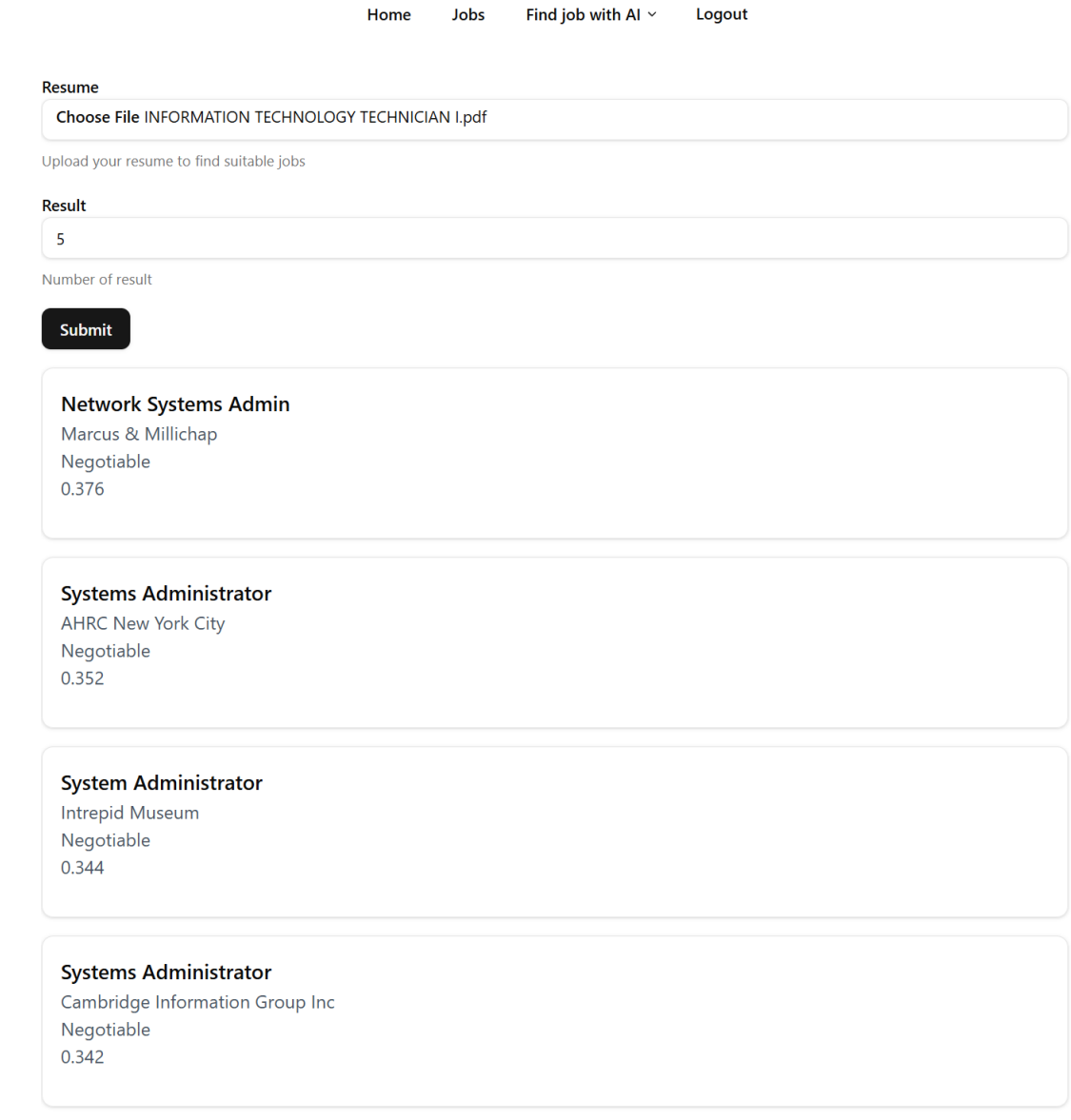


Hình 3.20: Màn danh sách công việc

Form upload CV:

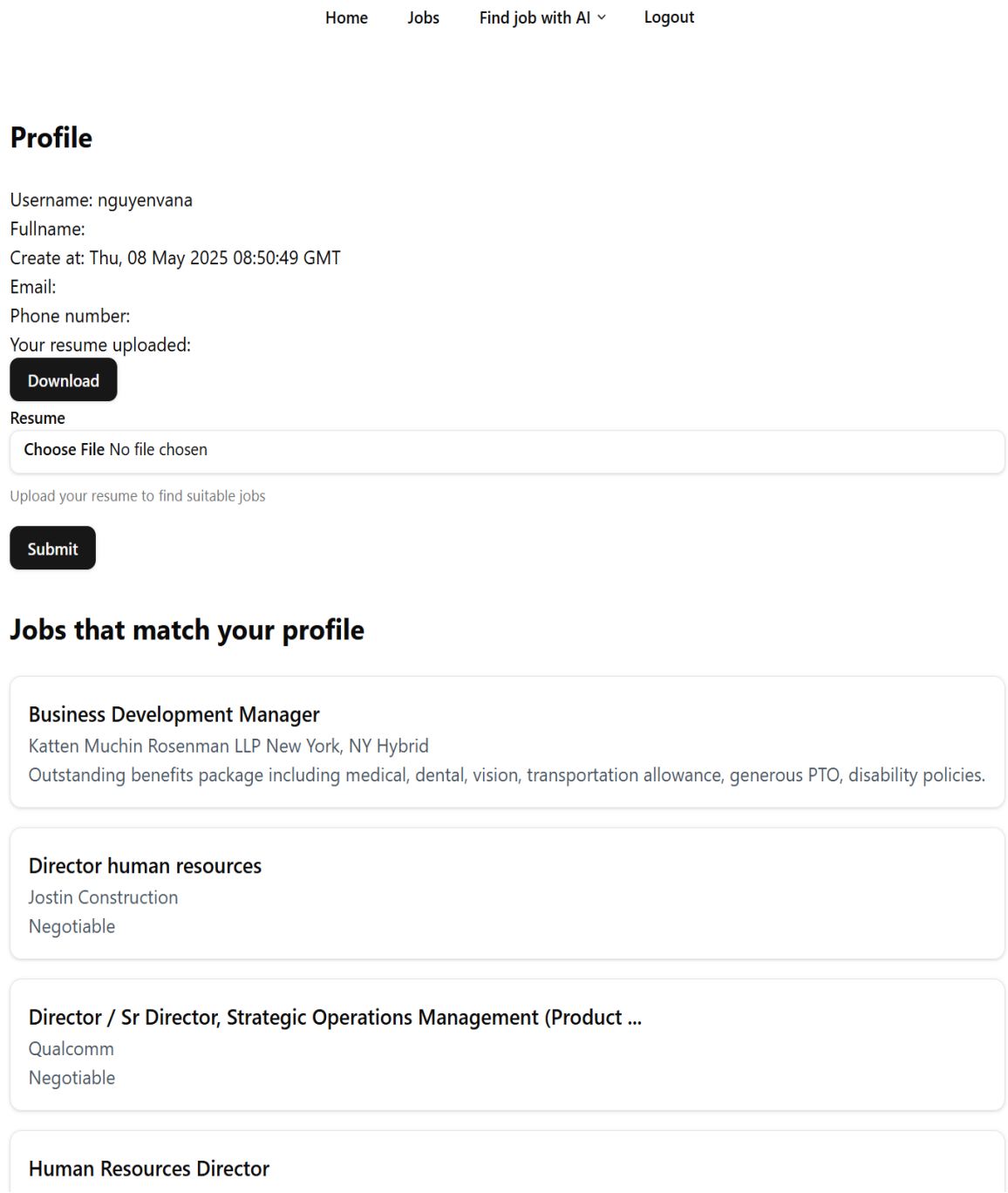
Hình 3.21: Form upload CV

Kết quả gợi ý việc làm theo form upload CV:



Hình 3.22: Kết quả gợi ý việc làm bằng công cụ

Kết quả gợi ý việc làm phù hợp với hồ sơ ứng viên:



Hình 3.23: Kết quả công việc phù hợp theo profile ứng viên

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1.Tóm tắt những kết quả chính

- Hệ thống gợi ý việc làm prototype đã được triển khai thành công với hai hướng tiếp cận: TF-IDF + KNN và BERT embedding.
- Phương pháp TF-IDF + KNN cho kết quả nhanh, với Precision@5 đạt ~0.68 và Recall@5 khoảng 0.54 trên tập thử nghiệm.
- Phương pháp BERT embedding cải thiện semantic matching, với Precision@5 đạt ~0.75 và Recall@5 khoảng 0.61, đồng thời đa dạng hóa kết quả tốt hơn.

2. Những đóng góp của đề tài

- Cung cấp phân tích chi tiết và so sánh hai mô hình phổ biến trong gợi ý việc làm: TF-IDF + KNN và BERT.
- Xây dựng quy trình end-to-end từ thu thập, tiền xử lý dữ liệu, đến triển khai API và giao diện web demo.
- Minh họa rõ ràng các bước tính TF-IDF, áp dụng KNN và tích hợp mô hình BERT pretrained với pipeline inference.
- Đề xuất các metric và phương án tối ưu hóa hệ thống (ví dụ: Elasticsearch k-NN, caching embedding).

3. Hạn chế của đề tài

- Dữ liệu thử nghiệm giới hạn trong vài nghìn bản ghi, chưa phản ánh đầy đủ quy mô thực tế doanh nghiệp.
- Hệ thống demo chưa tối ưu cho môi trường sản xuất (chưa có tính năng load balancing, monitoring, logging chuyên sâu).
- Chưa triển khai giải pháp explainable AI để minh bạch quy trình gợi ý, dẫn đến khó giải thích kết quả cho người dùng.
- Chưa hỗ trợ đa ngôn ngữ đầy đủ (chỉ demo với tiếng Anh).

4. Hướng phát triển trong tương lai

- Mở rộng dữ liệu: thu thập và làm sạch dữ liệu real-world lớn hơn (hàng chục nghìn JD và resume) để đánh giá độ ổn định của mô hình.
- Triển khai giải pháp explainability (LIME, SHAP) cho BERT để giải thích vì sao một job được gợi ý.
- Ứng dụng Federated Learning và kỹ thuật bảo mật (differential privacy) để bảo vệ thông tin cá nhân người tìm việc.
- Tối ưu hoá hạ tầng: sử dụng vector database chuyên biệt (FAISS, Milvus), triển khai microservices, autoscaling.
- Mở rộng hỗ trợ đa ngôn ngữ (tiếng Việt, tiếng Nhật, tiếng Hàn) và fine-tune mô hình BERT multilingual.

TÀI LIỆU THAM KHẢO

[1]. https://scikit-learn.org/stable/modules/neighbors.html

[2]. https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

[3]. https://www.kaggle.com/code/siddigantm/task-resume-matching-with-job-descriptions

[4]. https://huggingface.co/docs/transformers/model_doc/bert

[5]. https://viblo.asia/p/hieu-hon-ve-bert-buoc-nhay-lon-cua-google-eW65GANOZDO

[6]. https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11/