

Inclusive Language and Discrimination Detection in Job Descriptions

1. Introduction

This project is in collaboration with Deloitte company. In this project, we will implement a system which can detect any non-inclusive and discriminative language in job descriptions in Italian Language. Non-inclusive language are words or phrases that treat people unfairly, are insulting or exclude a person or a group of people. If this happens in the workplace, people may become silent, they may no longer feel accepted and part of the team. As an example:

Sei pronto a #makeanimpactthatmatters nel nostro team Program Governance & Major Programs?

is non-inclusive because it is using ['pronto'] in the text which implies that the ideal candidate is a man by using only masculine forms.

2. What have we done so far?

Step 1: Collecting seed data and labeling them by human expert

At the start of the project, some human experts in the law department helped us to collect job descriptions from LinkedIn and other platforms. This data is available in the following [link](#). The main columns in this dataset are:

1. **text:** contains the job description text.
2. **inclusive phrasing:** is the binary label of job description which is YES/NO.
3. **context:** is a reason which states that why the job description is labeled as YES/NO.

In the future, we will use inclusion category and alternate phrasing as well. Inclusion category determines the category of discrimination/exclusiveness that now, for most of the samples is Gender. Also, our designed system should suggest alternative phrasing which is neutral and inclusive for preventing non-inclusive language after detecting it.

Step 2: Improving the context column from the seed dataset

The initial version of seed dataset didn't include the reason for the inclusive language (YES labels). Also, for non-inclusive language (NO label), it used the same reason for all rows. For better results in fine-tuning and prompting we did an improvement in the context column which

1. Having reason for both labels.
2. Specifying tokens that were inclusive/ non-inclusive in the reason.
3. Having a structure for this reasoning which is label + tokens + reason.

As an example, for **YES** label we have:

This job description is **inclusive** because it is using ['pronto/a'] in the text which **covers both masculine and feminine genders**.

And for **NO** label we have:

This job description is **not inclusive** because it is using ['empatico', 'tuoi colleghi', 'degli stakeholder', 'tutto il team'] in the text which implies that **the ideal candidate is a man by using only masculine**

These improvements will help the LLMs to better understand the language and the differences between these two types of language. Also, it will help us to find out the inference that LLM used for labeling the data and we can have a better validation based on this context structure.

Step 3: Using modular prompting for better results

From an anecdotal experience, we know that different LLMs behave differently depending on the prompt's structure (Fig 1). To quantify this influence, we choose a modular approach for the classification prompt, composing it from these components:

- C: the context setup, always the first element
- I: the instructions for the task
- E: the evidence to classify
- Q: the question to classify for
- X: the closing instruction to provide the classification, or
- T: a *call-to-thinking* to “motivate” the LLM to consider the task carefully, including a mention of the question; an instruction to classify; and an ask to later provide an explanation for the classification. This prompt ending is an alternative to X.

Using these modular components, we generate prompts with the following component permutations (the trailing X is not mentioned, it applies for all prompts that do not end in T):

CIEQ, CIET, CIEQT, CIQE, CEIQ, CEQI, CQEI, CQIE, CIEQQ, CQEIQ, CQEIQ, CQIEQ

	CIEQ	CIET	CIEQT	CIQE	CEIQ	CEQI	CQEI	CQIE	CIEQQ	CQEIQ	CQEIQ	CQIEQ
gpt-35-turbo-1106	0,614	0,518	0,555	0,594	0,582	0,615	0,524	0,565	0,707	0,620	0,599	0,701
gpt-35-turbo-instruct	0,570	0,578	0,595	0,448	0,557	0,505	0,426	0,445	0,580	0,579	0,452	0,579
text-davinci-003	0,720	0,729	0,749	0,628	0,771	0,741	0,582	0,578	0,725	0,703	0,673	0,752
mixtral-8x7B-instruct-v01	0,804	0,771	0,811	0,676	0,806	0,695	0,580	0,623	0,769	0,758	0,692	0,771
claude-2.1	0,774	0,528	0,729	0,573	0,804	0,756	0,638	0,572	0,744	0,790	0,741	0,756
claude-instant	0,717	0,694	0,743	0,634	0,693	0,658	0,539	0,601	0,706	0,626	0,630	0,690
bison-002	0,843	0,791	0,836	0,658	0,823	0,808	0,671	0,642	0,737	0,764	0,729	0,764
unicorn-001	0,792	0,765	0,779	0,746	0,813	0,820	0,780	0,769	0,774	0,801	0,799	0,777
gemini-pro	0,833	0,812	0,796	0,662	0,839	0,800	0,669	0,661	0,766	0,772	0,749	0,762
command	0,747	0,685	0,714	0,548	0,722	0,746	0,599	0,537	0,707	0,731	0,715	0,732
gpt-4-1106-preview	0,874	0,853	0,839	0,775	0,857	0,835	0,734	0,757	0,831	0,824	0,798	0,848

Fig 1- F1 scores across models and prompt structures

For example, the CIEQ prompts look like this:

C: You are an accomplished AI working as a human resource expert in an Italian company.

I: Your task is to analyze some job description in Italian and identify any non-inclusive language based on Italian language-specific rules. This includes considering the different endings of adjectives and nouns based on masculine and feminine genders. Pay close attention to the details and context of each job description by Italian language rules.

E: This are the raw text from some job descriptions:

{job_desc} ===END===

Q: This is the question: {Are the following job descriptions inclusive or non-inclusive?}.

T: Respond only with the exact binary label of each job description and then start a new paragraph and explain why by specifying the exact tokens in the text of job description that lead you to the identified label.

Based on the above explanations, we are going to test different prompts for finetuning depending on the model that we want to use. So, we made an Excel file which consists of different ideas for each module that is accessible using this [link](#).

One of the approaches here is testing different roles (Context) like lawyer, journalist, student, HR and linguistic expert for the model. By using this approach, the model tries to think as the given role, and it gives us different aspects of the analysis and reasoning.

Step 4: Using Zero-shot learning (ZSL)

Zero-shot learning (ZSL) is a machine learning scenario in which an AI model is trained to recognize and categorize objects or concepts without having seen any examples of those categories or concepts beforehand. An example for zero-shot prompt is like this:

Classify the following job description in Italian to "inclusive" or "non_inclusive" language: {job_desc_list}

We used this strategy with GPT-3.5 and a pretrained model based on BERT and the results is shown below:

Model	Recall	Specificity	Precision	NPV	F1-score	Accuracy	Balanced accuracy
GPT-3.5	0.6491	0.3659	0.5873	0.4286	0.6163	0.5306	0.5075
mDeBERTa	0.965	0.293	0.655	0.857	0.781	0.683	0.629

Step 5: Using Few-shot Learning (FSL)

Few shot learning is a prompt engineering technique where you insert examples in your prompt, training the model on what you want the output to look and sound like. This method builds on LLM's ability to learn and generalize information from a small amount of data. This makes it particularly helpful when you don't have enough data to finetune a model.

An example for few-shot prompt in our case is like this:

C: You are a native Italian law and linguistic expert.

I: Your task is determining the label of some job descriptions to see whether they have "inclusive" or "non-inclusive" language. You must consider that the Italian language is a gender language which means that we have different endings for nouns and adjectives based on the gender we are talking about. So, for example, the following job description {Sei pronto a #makeanimpactthatmatters nel nostro team Operating Model Transformation FSI? } is "non-inclusive" in terms of "gender" because the job adv implies that the ideal candidate is a man by using only masculine forms, singular and plural.

Q: You must set a binary label as the output. Based on the above instructions classify the following job descriptions as "inclusive" or "non_inclusive":

E: {list_of_job_desc}

Using the above prompt with GPT-3.5 as the baseline gives us the following results:

Model	Recall	Specificity	Precision	NPV	F1-score	Accuracy	Balanced accuracy
GPT-3.5	0.964	0.292	0.654	0.857	0.779	0.683	0.629

But with an improved prompt and by specifying the tokens and structured reasoning like below we got better results in terms of specificity and precision:

C: You are an accomplished AI working as a lawyer and linguistic expert in an Italian company.

I & Q: Your task is to revise some job descriptions in Italian and identify non-inclusive tokens that refer to the candidate or working environment in the text. Then, you must identify the final label and the reason that led you to the identified label. For example, this job description {"positive_example"} labeled as "inclusive" because {"it uses [inclusive tokens] in the text which covers both masculine and feminine genders."}. As another example {"negative_example"} is labeled {"non_inclusive"} because {"it uses [non inclusive tokens] in the text which implies that an ideal candidate is a man by using only masculine forms."}.

T: You must keep the structure {"job_description"}: {"label"} + {"reason"} as the provided example.

Model	Recall	Specificity	Precision	NPV	F1-score	Accuracy	Balanced accuracy
GPT-3.5	0.4815	0.7556	0.702	0.5484	0.5704	0.6061	0.6186

Step 6: Finetuning Large Language Models (LLMs)

Fine-tuning is the process of taking a pre-trained model and further training it on a domain-specific dataset. Fine-tuning tailors the model to have a better performance for specific tasks, making it more effective and versatile in real-world applications. This process is essential for tailoring an existing model to a particular task or domain.

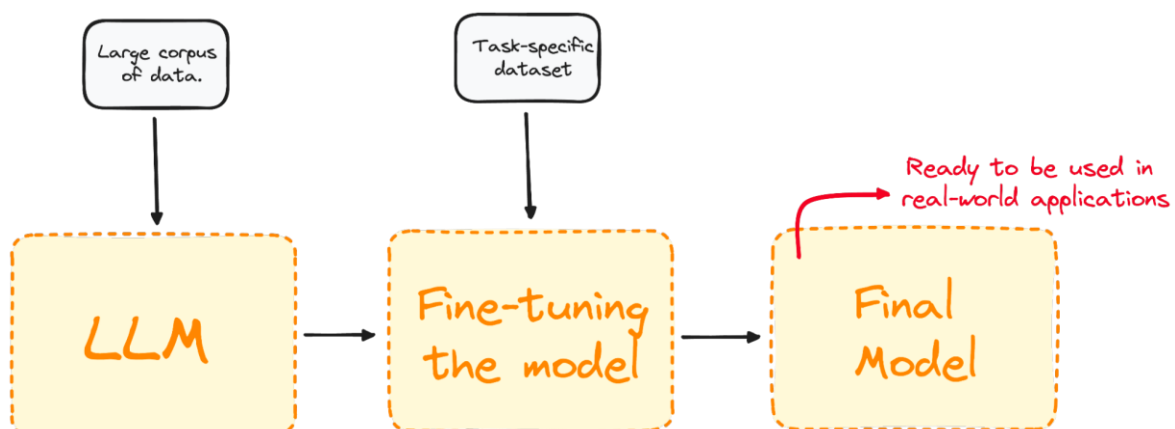


Fig 2- Finetuning process

There are different types of finetuning. For example, FSL is a type of finetuning, but in this case, by finetuning we mean training the model on a labeled dataset specific to the target task to perform.

One of the challenges that we face is the low amount of data in our seed dataset, so we need to augment our dataset using generative models or collecting more data.

Even though all fine-tuning techniques are a form of transfer learning, this category is specifically aimed to allow a model to perform a task different from the task it was initially trained on. The main idea is to leverage the knowledge the model has gained from a large, general dataset and apply it to a more specific or related task. We used this approach for finetuning the [Multilingual BERT](#) and below the result is shown:

Model	Recall	Precision	F1-score	Accuracy
MBERT	0.667	0.5	0.571	0.4

Step 7: Collecting more data using prepared APIs

As mentioned in the previous section, for solving the challenge of low data for finetuning we used a [prepared API](#), and we could collect about 1300 rows of data. Here is the [link](#) for collected data. Also, the job titles we searched for are listed [here](#).

3. What must we do in the future?

In the future, we must implement some LLMs and evaluate their performance for finalizing our system for use in real case. For this purpose, we must do the following steps:

Step 8: Initial labeling for collected data using API


For using this collected data in our training and finetuning we need to label them like our seed dataset. An effective approach for this step is using FSL with a good prompt to reach the most accurate results. So first we have to extend our prompts dataset and then we must test them using accessible models like GPT-3.5 or 4. When we reach a good prompt, we should use that for initial labeling our data. In the link below, you can find the results of initial labeling using some prompts with GPT-4.

[Revised_Job_Descriptions_v1.xlsx](#)

Also, we can use pretrained models like Llama for this step. We are working on this step.

Step 10: Validating the initial labeling by the help of human experts

Because we labeled our data using automated labeling by LLMs, we are not sure about the correctness of the labels. So, to ensure the labels and data quality used in finetuning step, we must validate the initial labeling using human experts' knowledge. For this purpose, we are going to extend one of our previous systems that is called **Annotix**. This system is an annotator which can help us to accelerate the validation process and make it easier. Figs 3 and 4 demonstrate a preview of the designed layouts of this system for labeling validation:


user

Instruction

1. Please read the text of job description and if the label is not correct change it using the **Change default label** button.
2. Also, if the reason is not correct by using **Edit reason** button you can edit it.
3. Please pay attention to the highlighted tokens in the text. If they are not correct or any tokens is missed please use the **Edit Tokens** button to fix it.

Job Description

Sei **pronto/a** a #MakeAnImpactThatMatters all'interno di Officine Innovazione? Officine Innovazione è la società del network italiano Deloitte che promuove la cultura dell'innovazione e fornisce alle imprese clienti servizi di innovation development e management. Essa opera a livello nazionale e internazionale, anche attraverso iniziative multi-player e partnership con attori chiave dell'ecosistema, con lo scopo di guidare l'evoluzione della business community in Italia in una logica più ampia di Trasferimento Tecnologico, Open Innovation e Venture Building, integrando a 360° il mondo delle aziende consolidate, delle startup/scaleup, del venture capital e delle università e centri di ricerca.

Edit Tokens

Default Annotation

Label

Inclusive

Change default label


Reason

This job description is inclusive because it is using ['pronto/a'] in the text which covers both masculine and feminine genders.

Edit Reason

Add your comment here....

Fig 3- designed layout for inclusive job descriptions in Annotix system


user

Instruction

1. Please read the text of job description and if the label is not correct change it using the **Change default label** button.
2. Also, if the reason is not correct by using **Edit reason** button you can edit it.
3. Please pay attention to the highlighted tokens in the text. If they are not correct or any tokens is missed please use the **Edit Tokens** button to fix it.

Job Description

Sei **pronto** a #makeanimpactthatmatters nel nostro team Operating Model Transformation FSI?

Edit Tokens

Default Annotation

Label

Non_Inclusive

change default label

Reason

This job description is not inclusive because it is using ['pronto'] in the text which implies that the ideal candidate is a man by using only masculine forms.

Edit Reason

Add your comment here....

Fig 4- designed layout for non-inclusive job descriptions in Annotix system

Step 10: Using the seed dataset + collected dataset for finetuning

In the meantime, we must start the finetuning process. For finetuning we are going to try following models using different prompting strategies as we mentioned in step 3:

1. BERT (and its different versions)
2. Llama (and its different versions)
3. Claude
4. Mistral
5. Gemini
6. GPT (if OpenAI API provided)

For this step it is important to record all the performance evaluation results like the examples shown in this document. Also, the details of parameters, optimizing, finetuning strategy and Parameter-efficient Fine-tuning (PEFT) method is important to record. We can share our code in the provided [GitHub link](#) for the project. In addition, the progress tracking of the project is possible using this [link](#).

Ongoing Tasks

Based on the above steps and details of the project, we have these task to be done:

Tasks	Description	Related Step
Prompts Generation	Generating more options for each module (C, I, Q, T, X) based on modular prompting approach for both English and Italian language and save the results in the shared document . 'E' will be our job descriptions so it's constant. Please consider different roles and different ways to ask Questions to have more variety in prompting.	Step 3
Prompt Testing	Choosing a structure for your prompt (from Fig 1) and testing different prompts (that you generated in the previous step) using GPT models and see which prompt gives us better results.	Step 5
Data Collection	Collecting more job description data using Rapid API by your account and using the option of job title (list of job titles that already searched is in this link) and country (Italy) search and saving the result as JOSN or CSV	Step 7
Initial Labeling	Using ZSL or FSL for initial labeling the data that collected using the API (using different prompts that gave you best results). The labeling should have a structure like the seed dataset i.e. we need a binary label (Inclusive/ non_inclusive), a reason and tokens that lead to the specific label. Keep the results here .	Step 8
Annotation Validation	If you're interested , you can help us in implementing the web page for annotation validation as we described in Step 10.	Step 10
Fine-tuning a LLM	Use the seed dataset + collected and labeled data for finetuning a LLM (the list of is described in Step 11). In this task, keeping the results of performance is so	Step 11

	important. Also, we need to record details of the parameters and methods that we used in finetuning. Save the results in the provided shared document .	
--	---	--

Feel free to reach me with any questions and we are open to your ideas for progressing in this project!

Fatemeh.mohammadi@unimi.it

References

- [1] <https://medium.com/@olaf.lenzmann/mastering-llms-for-complex-classification-tasks-64f0bda2edf3>
- [2] <https://www.datacamp.com/tutorial/fine-tuning-large-language-models>
- [3] https://medium.com/@dan_43009/few-shot-prompting-what-it-is-when-to-use-it-examples-limitations-and-biases-92947f0b88e1#:~:text=Few%20shot%20prompting%20is%20a,a%20small%20amount%20of%20data.
- [4] <https://medium.com/@dillipprasad60/qlora-explained-a-deep-dive-into-parametric-efficient-fine-tuning-in-large-language-models-llms-c1a4794b1766>
- [5] <https://arxiv.org/abs/2210.02406>
- [6] <https://medium.com/@ipopovca/the-magic-of-modular-prompts-in-gpts-ab832ce0f775>
- [7] <https://dassum.medium.com/fine-tune-large-language-model-llm-on-a-custom-dataset-with-qlora-fb60abdeba07>