

Università degli Studi di Pavia

# CLINICAL TRIALS SEARCH ENGINE

Information Retrieval & Recommender Systems

Michele Ventimiglia, Manuel Dellabona

# The Project



## System aim and usage

- Develop a search engine to retrieve relevant clinical trials based on written summaries of patients' medical conditions.
- The SE will be used by medic personnel to fasten up the triage of a patient basing on previous trials.

# Dataset exploration

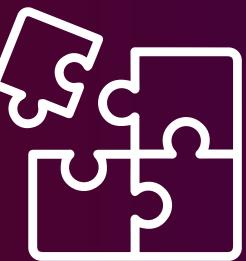


## What are we looking for?

- After analyzing the TREC CDS dataset we observed key features such as the condition, the description, and more (see a few below).
- We used all the data available in the document to build the best search engine we could.

nct_id	brief_title	brief_summary	detailed_description	condition
NCT00 977769	Carbetocin Versus Oxytocin and Hemodynamic Eff...	A randomized double-blind trial of oxytocin 5 ...	Healthy pregnant women scheduled for elective c...	Effects of Anesthesia, in Pregnancy

# Workflow



 First Idea

 Development

- Experiments → New Features → Test & debug 

 Deployment

# Intuition



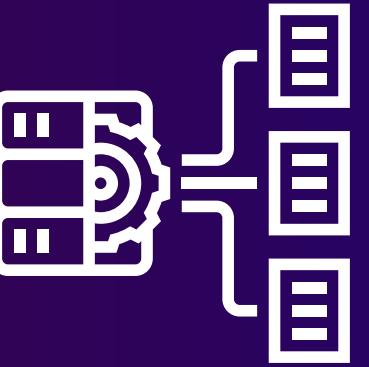
Which features can be useful for our users?

- Use a Large Language Model (LLM) to expand and upgrade the query.
- Implement a reranking sys based on user feedback can enhance the experience, since the search is supposed to read more than one study.



Clinic Trials SE - Michele Ventimiglia, Manuel Dellabona - University of Pavia

# Extraction & Parsing



How do we get the TREC CDS documents?

From **XML** files:

- Recursively extract
- Parse each xml

From **Pickle** file:

- Load the database

# Preprocessing & Indexing

How do we process the raw text from the documents?

Using spaCy we implemented the following steps:

1. **Punctuation Removal**
2. **Stop-Words Removal**
3. **Lemmatization**

Then we index all the processed documents using PyTerrier.

# Query Formulation



How do we manage the queries?

1. User Input → Processing → Expansion (PyTerrier) → Original Query
2. **Translation** → Processing → Translated Query
3. **PMC Extraction (LLM)** → Processing → Query + PMC

# Translation 文A

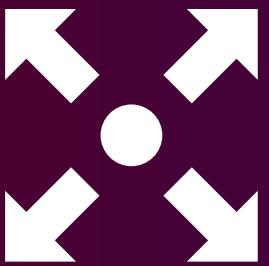
## Why translate?

It improves the query results and it can be language inclusive.

## Simple is better

- Using large language models or local solutions resulted in non optimal solutions.
- We chose to use an open **Google Translate Endpoint** to make it better.

# Expansion

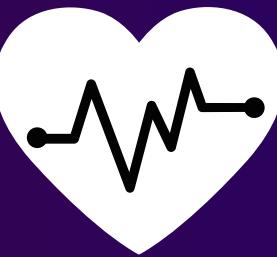


We both expanded the queries with PyTerrier default methods.  
In particular the best results were obtained with the methods applying  
Relevance Models and the Divergence from Randomness Framework.

## Reference Paper:

- Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. UMass at TREC 2004: Novelty and HARD. In Proceedings of TREC 2004.
- Amati, Giambattista (2003) Probability models for information retrieval based on divergence from randomness. PhD thesis, University of Glasgow

# LLM PMC Extraction

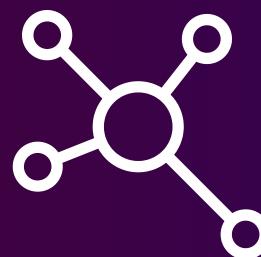


## Extracting Primary Medical Condition

Given a query we try to expand it extracting the PMC with a large language model. We use the following prompt instructions:

```
● ● ●  
1 instructions = (  
2     "Your must extract a medical condition from the given prompt."  
3     "If a medical condition is found, your answer must be just the medical condition."  
4     "You should not answer if no medical condition is found!"  
5     "Your answer must not include comments or opinions!"  
6 )
```

# LLM Digression

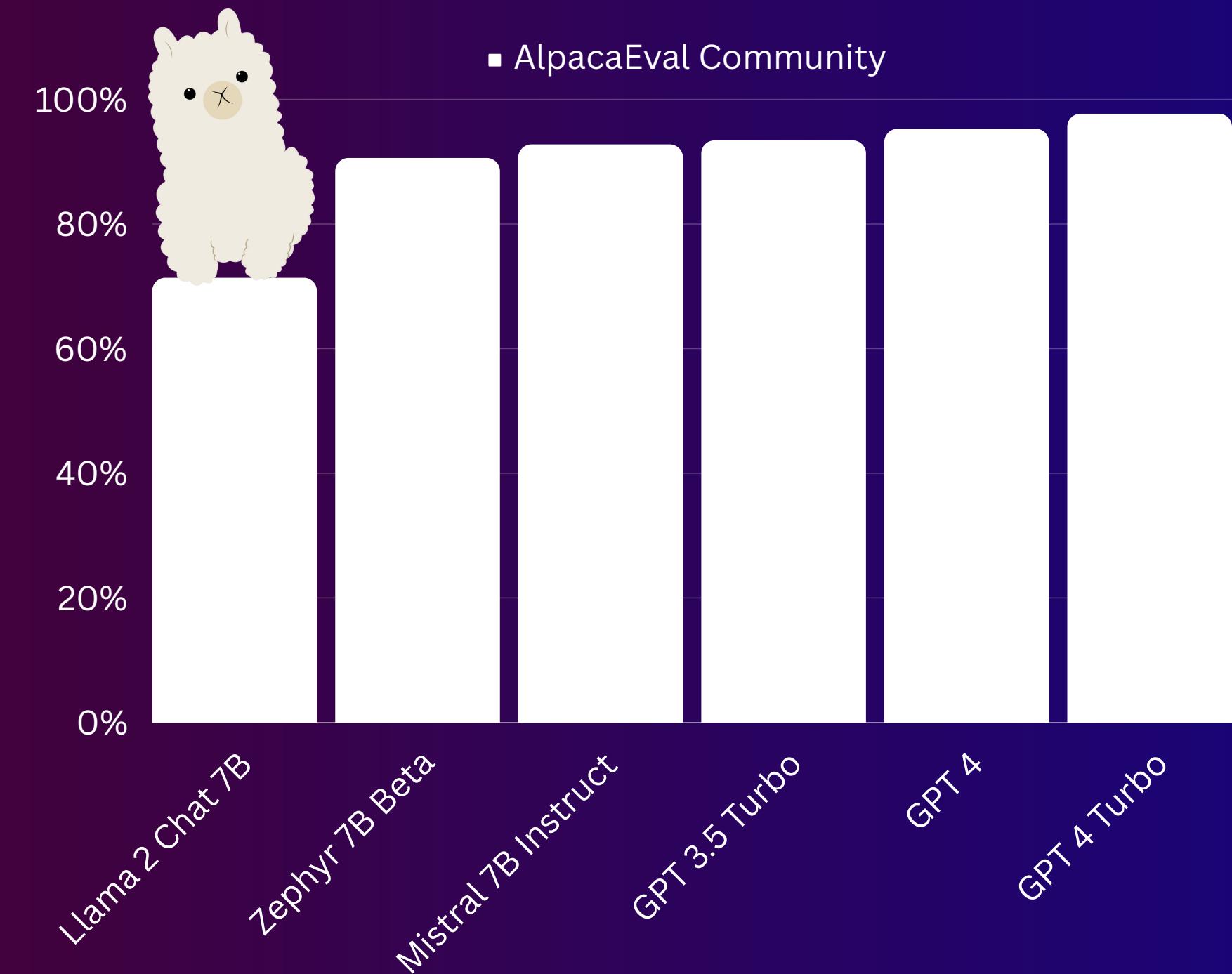


## Models

We implemented models such as:

- **Mistral 7B Instruct v0.2**
- **Zephyr 7B Beta**
- **Llama 2 Chat 7B**

These are not the best models overall but a good starting point on how to implement also the bigger ones.



# Retrieval & Ranking



## How do the documents are retrieved?

Given the union of all the processed queries we do the following steps:

- Using PyTerrier we first search and retrieve the ranked documents list using a baseline (BM25 or TF-IDF) and a default PyTerrier Query Expansion (not applied in case of LLM QE).
- We then rank the list using a policy for optimizing the combined results of the original, translated and the expanded query (optional).

# User Feedback & Settings

## How do we improve the System-Human Interaction?

- Through the Graphical User Interface we implemented the possibility for the user to tell the SE if the document retrieved is of relevant or not.
- The user can also decide to expand the query or extract the PMC when searching using a simple graphic implementation.

# Reranking

## How a neural similarity matrix can help in this task?

We implemented a fine-tuned BERT (ClinicalBERT on Huggingface) so that when the system receive a feedback for a document:

- A similarity matrix between the specified documents and all the other results is calculated.
- Based on a specific policy the results are reranked basing on the similarity between the documents

# Evaluation



## How to evaluate?

- **Topics:** the queries
- **qrels:** the correct results made by expertise to compare and evaluate the obtained results

## Metrics

- **Reciprocal Rank:**  $RR(\text{rel}=n)@X$ ,
- **Precision:**  $P(\text{rel}=n)@X$ ,
- **R-precision:**  $R\text{prec}(\text{rel}=n)$ ,
- **Recall:**  $R(\text{rel}=2)@X$

# Results

## Most relevant RS metrics

Blue: rel = 1 - Red: rel = 2

Retrieval System	RR@1k	P@1	P@10	Rprec	RR@1k	P@1	P@10	Rprec
TF-IDF	0.674	0.560	0.424	0.192	0.470	0.320	0.252	0.149
TF-IDF > RM3	<b>0.689</b>	<b>0.600</b>	0.468	0.247	0.458	0.340	0.270	0.179
TF-IDF > Bo1	0.677	0.580	0.488	0.254	<b>0.498</b>	<b>0.380</b>	0.314	0.191
TF-IDF > Llama 2	0.649	0.540	0.402	0.193	<b>0.489</b>	<b>0.380</b>	0.242	0.155
TF-IDF > Mistral	<b>0.686</b>	0.580	0.404	0.198	0.466	0.320	0.240	0.158

# Adaptations & Limits

## Models

- **LLMs** are not the best in terms of **determinism**, each result can be different.
- The lack of server / client implementation lead to a **slow startup and slow computing** basing on the device's performance.

## Live Feedback

- As the name suggest it is not permanent: a new query will erase all the user tuned results.
- The evaluation is not implemented since it would not be simple: it can be very expensive in term of resources and time!



# Live Demo

# Some Conclusions



## Neural Networks can be useful in IR

The inner nature of LLMs marks a clear limit between them (ex. chatbot) and search engines but NN can still improve the experience in using the search engines by being implied in some features.

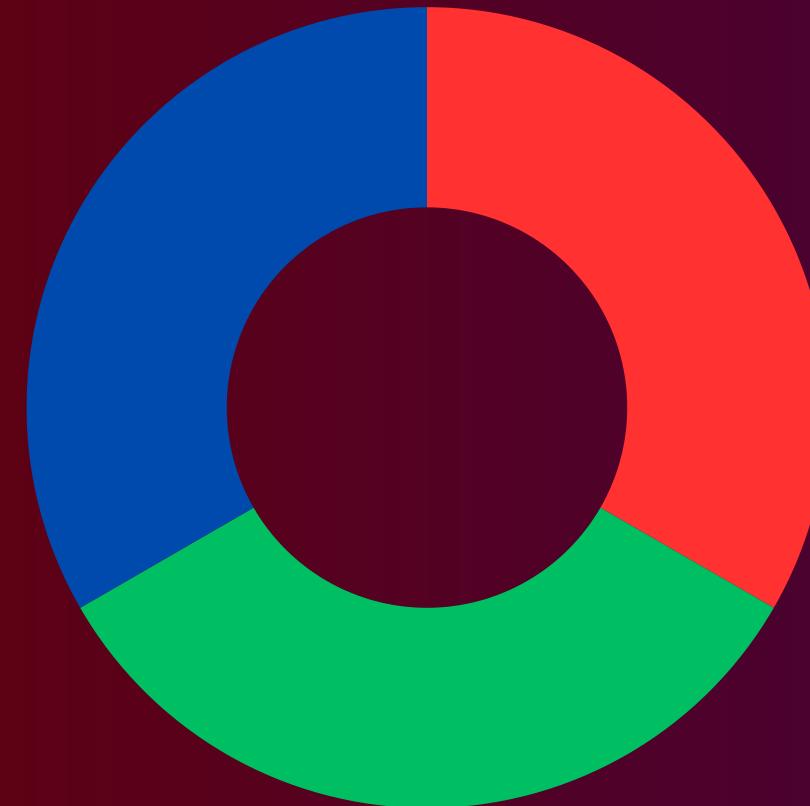
## Developing a Search Engine is not an easy task

A good project requires resources, time, multi-level and multi-disciplinary skills, back-end, front-end, data analysts and expertises in the SEO. However we managed to do it.

# What we gained



Experience, but also:



- ❑ Practical application of theory
- ❑ Understanding of LLMs
- ❑ Collaboration and Teamwork
- ❑ Problem-Solving experience
- ❑ User-Centric design focus
- ❑ Technical proficiency

# Questions?

# **Thanks for watching!**