

# Natural language processing for reviewing search results from scientific literature

## *El Procesamiento de Lenguaje Natural en la revisión de literatura científica*

Melanie Calero Sánchez<sup>1</sup>, José Carlos González González<sup>2</sup>, Isabel Sánchez Berriel<sup>1</sup>, Guillermo Burillo-Putze<sup>3</sup>, José Luis Roda García<sup>1</sup>

### Introduction

Natural Language Processing (NLP) is a discipline with a long history. It emerged in the 1950s as a subfield of Artificial Intelligence (AI) and Linguistics, with the goal of studying the problems derived from the automatic generation and comprehension of natural language. Although earlier works can be found, it was in 1950 that Alan Turing published an article titled "Intelligence", in which he proposed what is now known as the Turing test as a criterion of intelligence.<sup>1</sup>

Currently, there is a large amount of high-quality information available in specialized repositories across different fields of knowledge. However, when a specialist faces the task of searching for information, the relevance of the results must be determined by reading and analyzing the content of the documents. To simplify and make this process more efficient, numerous studies can be found on information retrieval in scientific texts using NLP techniques.

In the work by Tommasel *et al.*,<sup>2</sup> the authors present a web application for information retrieval enriched with a recommender system for Danish medical documents. The documents provided to users are determined by applying NLP techniques through lexical and semantic searches. It also guides the user in their search by recommending related queries and suggesting medical concepts that direct the search.

A comparison between different document representation techniques applied to filtering and labeling documents for evidence-based searches in medicine can also be found, showing promising results when algorithms based on language models are used.<sup>3</sup> The representation of queries through NLP techniques in both patient information retrieval systems and medical literature searches is analyzed in the work of Peikos *et al.*<sup>4</sup> These authors conclude that expanding queries with medical entities and other patient data improves results compared to searches formulated by medical advisors

when applied to clinical records, though not in the scientific literature.

Other fields in which advanced NLP techniques have been applied for information retrieval include the legal domain,<sup>5</sup> and even NLP *per se*, as demonstrated by the authors of the NLP EXPLORER tool<sup>6</sup> — a portal not only for document search, but also for indexing and visualizing query results. The interest in researching these techniques in the medical field is reflected in the CLEF eHealth Evaluation Lab, which includes a thematic section on information retrieval in medical scientific literature.<sup>7</sup> Thanks to the precision of information obtained through these techniques, reliable solutions can be built that function as highly specialized filters, significantly reducing the amount of information presented to the researcher. Furthermore, based on these, query points can be created for users with basic computer knowledge, providing them access to cutting-edge information retrieval technology.

Extracting knowledge from a corpus of documents automatically or semi-automatically is becoming possible. An increasing number of researchers from various disciplines are acquiring programming skills in specific computer languages. With a simple set of instructions and the support of specialized libraries, researchers can obtain information from documents practically and easily. Tools such as Jupyter Notebooks or Colab have facilitated code writing in different languages to perform various tasks, including literature retrieval and analysis.

In November 2022, ChatGPT appeared — a conversational system based on generative AI. In this case, document processing can be carried out without writing a single line of code, but instead through natural-language questions posed by the user to the system, which, after performing the relevant searches (information retrieval), provides a response formulated in natural language (response generation).

PubMed<sup>8</sup> is a free database that mainly

#### Author Affiliations:

<sup>1</sup>Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, Tenerife, Spain.

<sup>2</sup>Servicio de Informática, Universidad de La Laguna, Tenerife, Spain.

<sup>3</sup>Servicio de Urgencias, Hospital Universitario de Canarias, Universidad de La Laguna, Tenerife, Spain.

#### Corresponding Author:

Melanie Calero Sánchez.  
Dpto de Ingeniería Informática y de Sistemas, Universidad de La Laguna.  
Camino San Francisco de Paula, 19.  
38200, La Laguna, Tenerife, Spain.

#### E-mail:

melaniecalero.5@gmail.com

#### Article Information:

Received: 19-6-2024.

Accepted: 28-6-2024.

Online: 2-7-2024.

#### Editor in Charge:

Manuel Pardo Ríos.

includes the MEDLINE database of references and abstracts on life sciences and biomedical topics. The U.S. National Library of Medicine (NLM) at the National Institutes of Health maintains the database as part of the Entrez information retrieval system.

From 1971 to 1997, online access to the MEDLINE database was mainly through institutional facilities such as university libraries. PubMed, first released in January 1996, marked the beginning of private, free access to MEDLINE. The PubMed system was made freely available to the public in June 1997.

As of May 23<sup>rd</sup>, 2023, PubMed contains more than 35 million citations and abstracts, dating back to 1966, selectively to 1865, and very selectively to 1809. On the same date, 24.6 million PubMed records include abstracts, and 26.8 million have links to full-text versions (of which 10.9 million articles are available in full text and free of charge). In the last ten years, an average of nearly one million new records have been added each year.

In this paper, we will apply various natural language processing techniques to analyze PubMed publications and evaluate how they relate to our research topic: suicide among youth and machine learning algorithms. Our aim is to provide a set of NLP strategies, techniques, and tools to assist researchers in the initial phase of document review for a specific topic.

To this end, the main NLP strategies and techniques are presented first, followed by an example (in Section 4) demonstrating the use of some of these techniques in our review case. The next section presents new generative AI-based strategies and techniques to perform the same review process, illustrated with an example for better understanding. Subsequently, we describe and implement current chatbot systems that facilitate interaction between the researcher and the corpus of articles retrieved through question–answer systems.

## Endpoints

The primary endpoint of this work is to demonstrate how the use of natural language processing facilitates and improves the collection of scientific literature. Specifically, it aims to:

- a) Analyze how NLP techniques assist in retrieving scientific literature.
- b) Describe the main NLP methods and strategies applicable to literature search and selection.
- c) Provide practical examples of the steps to follow in literature reviews using NLP.

## Classical natural language processing techniques

NLP is the field of AI dedicated to studying how machines can communicate with humans through natural languages such as Spanish, English, or Chinese. Any human language can, in theory, be processed by computers. Logically, economic or practical limitations mean that only the most widely spoken or digitally used languages have active computational applications.

NLP encompasses algorithms for both language understanding and language generation. These algorithms are used in software applications for specific purposes. Some require only comprehension, others are generative tasks, and others combine both components.<sup>9</sup>

Some examples of NLP applications include:

### Text categorization

Text categorization generally assigns a label or category to a document or text fragment. Depending on the purpose of the classification, this process may involve sentiment analysis, spam detection, topic identification, or simply categorization according to other free-form criteria.<sup>10</sup>

Sentiment analysis determines the sentiment (positive, negative, or neutral) expressed in a text. For example, “I loved the movie!” should be assigned a positive sentiment. More complex algorithms evaluate the sentiment expressed in a text regarding specific aspects.<sup>11</sup> In this case, if the aspects to evaluate are “restaurant” and “menu,” then in the expression “The restaurant was expensive, but the menu was cheap” the words *restaurant* and *menu* should receive negative and positive labels, respectively.

Regarding topic extraction, it is possible to automatically determine whether an article is about epidemiology, pediatrics, or another subject. These algorithms are particularly important for information retrieval.

### Automatic summarization

The techniques included in this category aim to summarize a text document or a set of documents, presenting the user with the main information in a concise and comprehensive manner. The primary goal is to generate, from a specific document, a text that captures the main ideas with minimal repetition.<sup>12</sup>

### Information extraction

Information extraction refers to the set of techniques that identify mentions in a text of semantic concepts or entities, as well as relationships between them.<sup>13</sup> The entities and relationships to be extracted depend on the domain of the text. For example, in a general domain, entities may include people, organizations, and places, while relations could include married to, employed by, or lives in. In a biomedical domain, entities might include protein, disease, and treatment, with relationships such as protein–protein or disease–treatment. At a more complex level, representing this information as a graph model allows the generation of knowledge graphs, which constitute conceptual maps of implicit information across document collections.

### Question answering

The goal of question-answering (QA) systems is to provide relevant answers to user questions expressed in natural language. The answers are generated from information contained in the system’s knowledge base, typically

created from documents relevant to the application's domain. The architecture of these systems includes three modules: question analysis, retrieval of relevant fragments, and answer generation.<sup>14</sup>

### Machine translation

NLP enhances machine translation systems such as Google Translate. These systems align words across languages, taking into consideration context and idiomatic expressions.

### Conversational agents

A conversational agent is an application that produces human-like conversations.<sup>15</sup> Its main task is to generate natural language responses to questions posed by a human, also in natural language. Other names for these systems include chatbots, virtual assistants, interactive agents, and digital assistants. Some systems are rule-based, analyzing the user's input and constructing a response using pre-defined templates. Others are deep learning-based neural systems. Among these, retrieval-based agents generate responses by computing the most relevant match, while generative conversational agents construct responses word by word, choosing at each step the vocabulary term that best fits, until a complete response is formed. There are also hybrid systems that combine retrieval and generation.<sup>16</sup> An example of a generative conversational agent is ChatGPT.

Several linguistic analysis levels are applied in NLP. Not all analyses are required for every NLP task — it depends on the application's objective:

1. Morphological or lexical analysis. Analyzes the internal structure of words in sentences to extract lemmas, inflectional features, and compound lexical units. It is essential for determining the basic syntactic category and lexical meaning.

2. Syntactic analysis. Analyzes the structure of sentences according to a grammatical model (logical or statistical).

3. Semantic analysis. Provides sentence interpretation once morphosyntactic ambiguities are removed.

4. Pragmatic analysis. Incorporates contextual analysis into final interpretation. This includes figurative language (metaphor and irony) and domain-specific world knowledge required to understand specialized texts.

Morphological, syntactic, semantic, or pragmatic analysis is applied depending on the application's goal. For example, a text-to-speech converter does not require semantic or pragmatic analysis, while a conversational system requires detailed contextual and domain information.

Some of these levels are further subdivided into tasks, some of which depend on others. The sequence of tasks required in each application is called a pipeline. The most common tasks include:<sup>17</sup>

1. Preprocessing: The preliminary step before any text processing. It includes cleaning tasks such as removing URLs and hashtags, converting to lowercase, etc. The aim is to eliminate noise from the text. It may also include standardization tasks such as punctuation removal or stemming (reducing words to their root form, e.g., runs →

run). These steps ensure a cleaner input for subsequent NLP tasks.

2. Language identification: Each language has its own characteristics that, in certain analyses, will require specific processing. This is the first analysis that must be carried out, as it will, for example, help determine which data files such as dictionaries will be used in processing.

3. Tokenization: Tokenization is a key task in NLP. It involves dividing a text into smaller units, which can be words, punctuation marks, or sub-words. Depending on the intended application of the tokenizer, it may be important to include punctuation marks — for example, in sentiment analysis, exclamation marks "!" are relevant. The inclusion of sub-words solves problems of missing inflected word forms in test sets.<sup>18</sup> Given the sentence: "Fui diagnosticado con deficiencia de vitamina B12.", a tokenizer produces: ["Fui", "diagnosticado", "con", "deficiencia", "de", "vitamina", "B12", "."].

4. Sentence segmentation: Just as it is necessary to determine the smallest unit that algorithms will process through tokenizers, for many NLP problems it is also essential to identify the sentences in the text. This task is carried out by sentence segmentation algorithms.

5. Lemmatization: This is the process that converts words to their lemma or canonical form. For example, "investigado", "investiga", and "investigue" are inflections of the lemma "investigar." Lemmatization can be performed through morphological analysis or by using simpler stemming algorithms that extract only the root of the word.

6. Named Entity Recognition (NER): Identifies entities, typically names of people, organizations, or locations in the text. For example: Input: "Apple Inc. tiene su sede en Cupertino." Output: "Apple Inc. (ORG) tiene su sede en Cupertino (LOC)."

NER algorithms can also be applied to extract other types of entities, as in,<sup>19</sup> where a tool for German is presented whose algorithms can identify entities such as "medication," "concentration," "route," "form," "dose," "frequency," and "duration," which are relevant for understanding texts that include medical treatments. NER is crucial for information extraction and question-answering systems.

7. Part-of-speech tagging (POS): POS tagging assigns grammatical labels (such as noun, verb, adjective) to each word in a sentence. It helps in syntactic analysis and disambiguation. [Figure 1](#) shows the result obtained by *Freeling*<sup>19</sup> for the text "Yo me siento herida, mucho dolor en mi corazón."

8. Syntactic analysis: Extracts the syntactic structure of sentences.

9. Dependency parsing: Determines the grammatical relationships between the words in a sentence — subject, direct object, circumstantial complement. These are usually represented graphically, connecting related words and indicating with a label the type of relationship.

For example, for the sentence "La investigación es costosa pero aporta grandes beneficios," the result is shown in [Figure 2](#).

Yo me siento herida , mucho dolor en mi corazón  
yo me sentir herida , mucho dolor en mi corazón  
PP1C PP1C VMIP1S NCF500 Fc DI0MS0 NCMS SP DP1CS VMIS3P0  
SNO S00 0 0 000 S

Figure 1. Discourse analysis of the dataset.<sup>20</sup>

10. Coreference resolution: Identifies text fragments that refer to the same entity. For example, in the text: "Fui diagnosticado con deficiencia de vitamina D, el médico me lo diagnosticó. El doctor me la recetó," the coreferences are highlighted as follows: "lo" → "deficiencia de vitamina," "la" → "vitamina D."

The sequence of tasks to be applied depends on the specific problem and the typology of the language. In some languages, word order is highly relevant, while others are morphologically complex. The configuration of the pipeline and the algorithms applied will yield better results if they are adapted to the characteristics of the language. For example, English is not morphologically complex, so in that language the basic unit can be the word rather than the lemma or the root. Some languages tend to omit pronouns, such as Spanish; however, this is not the case in English, so in Spanish, coreference resolution is particularly important.<sup>17</sup>

The result of applying these analyses is text enriched with information regarding the structure and semantics of language. Machine learning algorithms are then applied to these texts to perform classification, information extraction, or any other target application. Learning algorithms can be supervised or unsupervised, depending on whether example data with known categories or predictions are available. In the case of language, enormous quantities of examples are needed to build a representative sample of the infinite possibilities of expressions produced by speakers of a language to ensure learning that leads to the generalization of information.

Currently, due to advances in computational power, it is possible to apply highly complex neural networks to information volumes encompassing vast amounts of documents. The models generated are known as large language models (LLMs), which possess a high degree of generalization in their comprehension and language-generation capabilities. For example, the GPT-3 model was trained with Common Crawl (a data repository of all web pages), WebText2 (web data obtained by OpenAI), Books1 and Books2 (book repositories), and all English Wikipedia articles, totaling approximately 500 billion tokens.<sup>21</sup> Training these models entails high costs in terms of money, energy, and time. Therefore, pretrained models are adapted to specific domains or tasks when the general structure captured in them is not adequate for the register, terminology, or objective of the learning task. The procedure for adapting these neural networks to our own texts is known, in the field of LLMs, as fine-tuning.

Language models, during training, learn contextual representations of words, transforming them into numerical vectors. They excel at tasks such as text completion, ques-

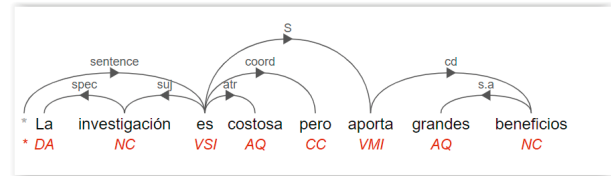


Figure 2. Dependency analysis obtained using the Freeling online demo.<sup>19</sup>

tion answering, and text generation. Among the best-known models are BERT<sup>21</sup> and GPT.<sup>22</sup> Model training refers to the computation performed by algorithms to adjust the probability distribution of word sequences, seeking to maximize the probability of sequences according to the text collections or corpora used for training. The Transformer architecture of these neural networks has represented a substantial leap in the quality of generated models, providing a high degree of reliability to NLP applications.

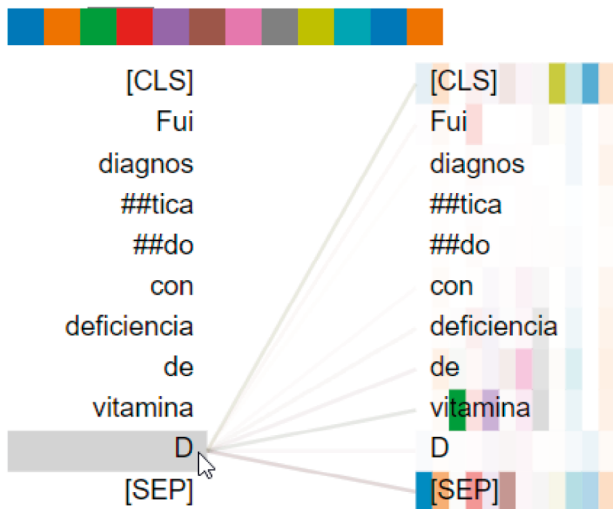
Example: "Once upon a time there was a brave masked knight called [MASK]." (BERT fills in the masked word).

Intuitively, we are mapping written language into a vector space, where different properties are represented as real numbers. In this field, the following fundamental concepts stand out in profiling the underlying technology:

1. Sequence-to-sequence models: These are neural networks that handle input and output sequences of variable length, with the goal of predicting unknown words in the input that are generated in the output. The predictions may refer to translation into another language, completion of the next part of a sentence in the same language, or filling in missing words in a text fragment. Recurrent Neural Networks (RNNs) and Transformer-based models fall into this category.

2. Attention mechanism: Introduced by the Transformer model, attention allows the model to focus on the most relevant parts of the input sequence. It computes values indicating, for a given word, the importance of the remaining words in understanding the sequence.<sup>23</sup> In Figure 3, these values are represented by the thickness of the lines connecting the word "D" in the sentence "Fui diagnosticado con deficiencia de vitamina D" using the Spanish language model bert-base-spanish-wwm-cased.<sup>24</sup> Notably, tokenization fragments the word "diagnosticado" into the tokens "diagnos," "tica," and "do." The token "vitamina" is especially important for understanding the meaning of the token "D."

3. Word embeddings: Word embeddings represent words or documents as dense vectors in a continuous space, referring to the numerical vector space in which language is represented. Popular methods include Word2Vec, doc2vec, GloVe, and FastText for obtaining semantic transformations. Transformers also generate vector representations that improve upon previous methods due to their ability to identify different senses of polysemous words. Current NLP systems that use machine learning techniques typically include the generation of these embeddings in their pipeline. They are particularly useful for capturing var-



**Figure 3.** Attention computed by the bert-base-spanish-wwm-cased language model for the word “D.” Visualization generated with BertViz.<sup>25</sup>

ious semantic relationships — for example, the analogy:  $\text{vector}(\text{“king”}) - \text{vector}(\text{“man”}) + \text{vector}(\text{“woman”}) \approx \text{vector}(\text{“queen”})$ . **Figure 4** shows embeddings of several words generated with Word2Vec<sup>26</sup>.

4. Textual similarity: Textual similarity provides an indicator of how similar two texts are, either in their writing or in their meaning. Orthographic similarity is usually calculated using Levenshtein distance, which counts the number of changes required to transform one word into another — the greater the distance, the lower the similarity. For example: “inflamación” and “influenza” have a Levenshtein distance of 5, whereas “inflamar” and “inflama” have a distance of 1. Semantic similarity, on the other hand, indicates how similar the meanings of two words are. This concept goes beyond synonymy, encompassing all types of semantic relations, such as analogy, semantic affinity between words, hyperonymy, which verbs predicate a noun, and many others. There are different approaches to calculating semantic similarity:

4.1. Similarity measures using lexical resources such as thesauri or ontologies. The similarity between two words is measured based on the path that must be traversed in the network to reach one word’s node from another in a semantic network such as WordNet or EuroWordNet.<sup>28</sup> WordNet is a semantic network of the English language in which words are organized into synonym sets called synsets. The network nodes (synsets) represent concepts that are linked through semantic relationships such as hypernymy, hyponymy, metonymy, holonymy, and antonymy, represented as network edges.<sup>29</sup> EuroWordNet is the version for a set of European languages, establishing interlingual relations between the synsets of each language. The structures for each language are included in the corresponding EuroWordNet<sup>30</sup> version, which also incorporates a common ontology across all languages and relations among them.

4.2. Language models, on the other hand, can generate semantic numerical spaces through word embeddings, in



**Figure 4.** Representation of a set of words in a three-dimensional numerical space using word embeddings obtained with the Word2Vec algorithm.<sup>27</sup>

which semantically related words are close to each other. Therefore, similarity measures can be computed based on the distance separating their embeddings, most commonly using cosine similarity. In **Figure 5**, the angles between the embeddings of the words “inflamación,” “gobierno,” and “república” are shown. The cosine of each angle corresponds to the similarity between the words: *inflamación–gobierno* (red angle) and *gobierno–república* (blue angle). The smaller the distance and angle, the closer the cosine is to 1, and thus, the greater the similarity.

The concept of semantic similarity is a fundamental component in NLP applications such as information retrieval, question–answering systems, text classification, and automatic summarization.

### Bibliographic retrieval using classical NLP techniques

In this section, we apply several of the previously mentioned techniques to the bibliographic retrieval of research on youth suicide in the PubMed document base. The code snippets shown for each technique are meant only to illustrate that extensive programming knowledge is not required. Notebook tools, such as Jupyter Notebooks or Google Colab, allow direct execution of these instructions once the relevant data are provided.

The goal of this example is to show how NLP techniques can speed up certain tasks in the literature review process. Specifically, the steps are:

1. Search all PubMed articles whose title contains the word “suicide.”

PubMed provides an easy way to export your searches to a preconfigured CSV file. Searches allow filters such as publication year, journal, and even keyword search.

In our case, we searched for publications since 2015 containing the word suicide, exporting a CSV with the fields ID, title, author, journal, publication year, and DOI.

(For readability in the remainder, words and phrases are shown in Spanish.)

2. Preprocess and tokenize the title to obtain individual words (tokens).

As described earlier, we follow a general pipeline for NLP. This step includes steps 1 and 3 of the pipeline. We convert all titles to lowercase, remove quotes and symbols such as @, #, etc., and from this preprocessed text we tokenize to obtain each word in the title.

3. Remove common, non-informative words (e.g., “y”, “el”, “de”) to focus on relevant keywords (stop-word removal). This cleaning step is helpful here but not always necessary. For document filtering by word similarity, we will need to vectorize and count occurrences. Removing repetitive, low-value words is key to avoiding distortion in the results.

4. Reduce words to their root form to improve search consistency (lemmatization).

This corresponds to step 5 of the pipeline detailed in Section 3. After applying it, all titles containing “suicidio” are tokenized and lemmatized.

5. Manually select words to relate to “suicidio” that should appear in the processed PubMed titles. For our use case, we select frequent, informative words in a subset of relevant publications, obtaining: *adolescentes* | *jóvenes* | *estudiantes* | *tendencias* | *medios* | *niños* | *escuelas* | *máquinas* | *redes* | *adolescencia* | *internet* | *tendencia* | *usuarios* | *móviles* | *redes* | *twitter* | *artificial* | *nota*.

For broader coverage, these are expanded with synonyms from EuroWordNet (Spanish version). The following Python snippet shows this task:

```
Code:
# Step 1: Initial keyword list
keywords = ['suicidio', 'jóvenes', 'adolescentes', 'prevención', 'factores de riesgo', 'intervención']
# Function to find synonyms using WordNet
def get_synonyms(word):
    synonyms = set()
    for syn in wn.synsets(word, lang='spa'):
        for lemma in syn.lemmas(lang='spa'):
            synonyms.add(lemma.name())
    return synonyms
get_synonyms(word='adolescente')
Output:
{'adolescente', 'chavea', 'imberbe', 'joven', 'juvenil', 'mozo', 'muchacho', 'pollo', 'zagal'}
Expanding the keyword list with WordNet:
# Step 2: Expand the keyword list
expanded_keywords = set(keywords)
for keyword in keywords:
    synonyms = get_synonyms(keyword)
    expanded_keywords.update(synonyms)
print(f"Expanded keywords: {expanded_keywords}")
Output:
{'cautela', 'factores de riesgo', 'operación', 'cuidado', 'adolescentes', 'suicidio', 'precaución', 'prevención', 'intervención', 'jóvenes', 'previsión'}
```

6. Check which manual words appear in the PubMed titles.

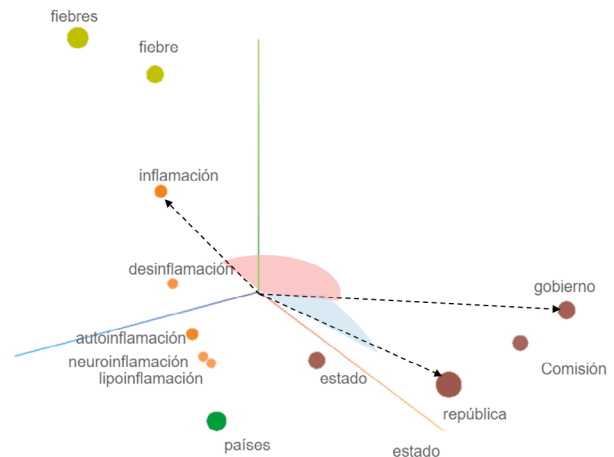


Figure 5. Angles formed between the embedding vectors of the words *inflamación*, *gobierno*, and *república*.<sup>27</sup>

This can be done via exact matching of the relevant texts or by using similarity metrics that provide a quantitative measure of word similarity. If we choose exact matching, a visual example can be shown as in Figure 6. For counting, one can use scikit-learn’s CountVectorizer, a well-known library in machine learning.

7. Using a similarity-based strategy, convert the tokenized titles from step 4 into frequency vector collections (numbers) to compute semantic distance. As discussed in Section 3, this is where we apply textual similarity techniques—specifically step 10 of the pipeline. We have each title tokenized and our keyword list; we then compute similarity between the title’s tokens and our keyword list to detect their presence.

Let’s see an example.

If we have the following titles:

```
titles = [
    "Suicidio en adolescentes: factores de riesgo y prevención",
    "Estrategias de intervención para prevenir el suicidio juvenil",
    "Análisis de casos de suicidio en jóvenes",
    "Prevención del suicidio en adolescentes en riesgo",
    "Factores psicológicos en el suicidio de jóvenes"
]
```

Vectorize to create the word-frequency matrix:

```
Vectorization using Bag of Words
vectorizer = CountVectorizer()
bow_matrix = vectorizer.fit_transform(filtered_titles)
Use cosine distance explained in Section 3:
Cosine similarity
cosine_similarities = cosine_similarity(bow_matrix)
# Create a DataFrame to visualize the similarities
cosine_sim_df = pd.DataFrame(cosine_similarities,
index=filtered_titles,
columns=filtered_titles)
# Visualize the cosine similarity matrix
print(cosine_sim_df)
# To obtain the titles most similar to a specific title:
```

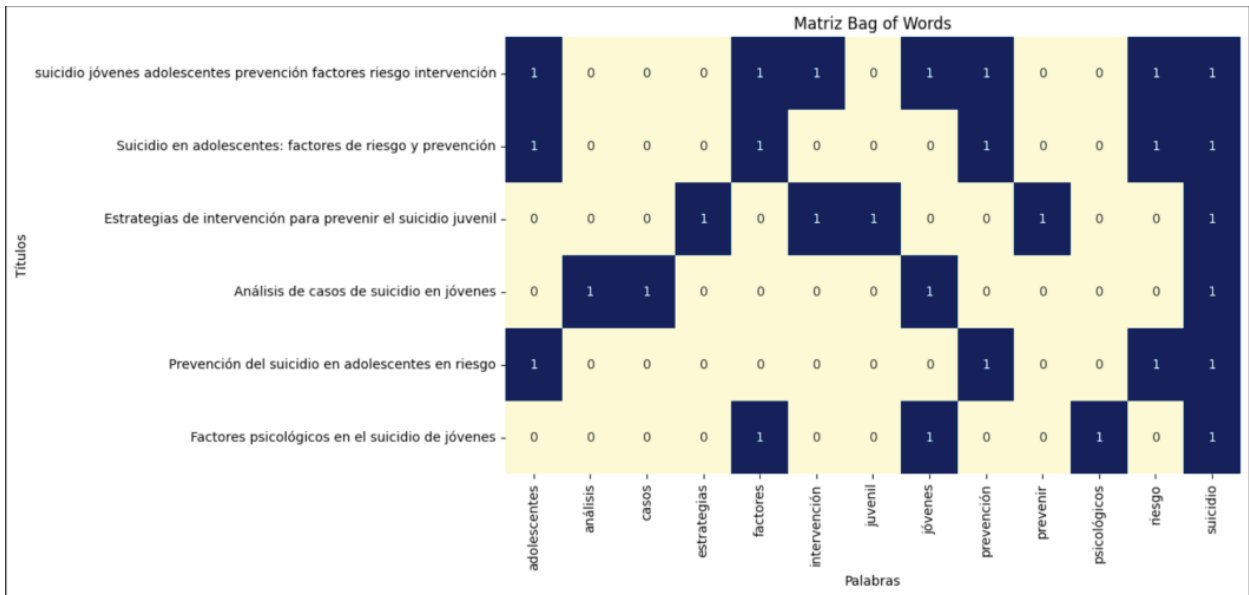


Figure 6. Example Bag of Words matrix illustrating exact word counting.

```

title_to_compare = "suicidio jóvenes adolescentes
prevención factores riesgo intervención"
similar_titles = cosine_sim_df[title_to_compare].sort_
values(ascending=False)
print(similar_titles)

```

We obtain what is shown in Figure 7.

As we can see, cosine similarity indicates how similar the manually selected words are to the sentences in the publication titles. The higher the number, the greater the similarity. We typically select those above a preset threshold.

8) From the publications obtained in point 7, we are interested in those related to "AI Model" and "Machine Learning." We again calculate the distance of these two terms individually and compute the similarity with the phrases of all the titles obtained in step 7.

9) Finally, we do the same for "nota suicidio." In summary, the pipeline or process flow followed in this example reduces the initial 9,212 publications containing the word "suicide" to the 74 final publications that also include the manually selected, relevant words. Figure 8 shows the pipeline or process flow.

### Literature review using generative artificial intelligence

Advances in generative AI (GAI) have made the process described in the previous section—which involves arduous data cleaning and the use of several advanced machine learning models—now as simple as asking a conversational system such as ChatGPT, Llama, or Gemini a question. A GAI system receives a user query, searches information in the corpus on which it was trained, and is able to generate a natural-language response understandable to a human. One of the most important components is the instructions or prompts, to the point that an entire discipline—prompt engineering—has emerged, consisting

of strategies and techniques for asking questions of the AI system. In this section, we show a simple example related to our literature retrieval on youth suicide.

Let us look at a case with different prompts that maintain coherence among the questions asked sequentially (this is what we call context). The examples are similar to those presented in the previous section but make use of the ChatGPT system:

*User: If I attach a file of publications, can you tell me which ones are related to youth suicide where the words suicide, note, artificial intelligence, and machine learning appear?*

*ChatGPT: Of course, I can help with that. Please attach the file of publications, and I will proceed to search for mentions of keywords related to youth suicide—specifically looking for the words suicide, note, artificial intelligence, and machine learning.*

*Once the file containing the publication titles has been attached, the conversational agent's response is as follows:*

*ChatGPT: Thank you for providing the list of publications. I will now review the titles to identify those related to youth suicide that contain the keywords suicide, note, artificial intelligence, and machine learning.*

*However, it seems that none of the provided entries explicitly contain the words artificial intelligence or machine learning. Even so, I will list the publications focusing on youth suicide:*

32699044 – "Access to Firearms and Adolescent Suicide"

35156223 – "Adolescent Suicide Attempts in the United States: When Suicidal Ideation and Capability Interact."

30771807 – "Suicide Among Indigenous Youth: A Crisis Without Relief"

36822834 – "Cyberbullying and Adolescent Suicide"

*None of these publications mention note, artificial intelligence, or machine learning in their titles, but they fo-*

suicidio jóvenes adolescentes prevención factor...	1.000000
Suicidio en adolescentes: factores de riesgo y ...	0.714286
Estrategias de intervención para prevenir el su...	0.267261
Análisis de casos de suicidio en jóvenes	0.251976
Prevención del suicidio en adolescentes en riesgo	0.503953
Factores psicológicos en el suicidio de jóvenes	0.428571

Figure 7. Cosine similarity matrix among analyzed publication titles.

cus on the topic of youth suicide. If you require a more detailed or specific search within the full text of these publications, I would need access to the complete content of the articles.

If you can provide more details or additional text fragments, I will be happy to assist you further.

Now, the system can request more information about the publications to be analyzed:

ChatGPT: Thank you for providing these additional titles. Here are publications related to youth suicide that mention suicide, note, artificial intelligence, and machine learning:

31079565 – “Using Machine Learning to Identify Suicide Risk: A Classification Tree Approach to Prospectively Identify Adolescent Suicide Attempters.”

This publication fits your criteria perfectly, as it mentions suicide, machine learning, and focuses on young people (adolescents). If you need more details about this specific publication or any additional information, please let me know.

We can see how the user can make requests easily, regardless of their level of computer knowledge. We can also observe that the responses will be consistent with the collection of documents available to the system. Moreover, the information returned will be more precise and reliable the greater the context we provide. These types of requests made to conversational agents based on LLMs are called prompts, and they can range from a simple direct question without further information (for example, “I want a collection of texts about suicide and machine learning”) to more specific ones that add context:

“I need to know, from this collection of files containing scientific articles, which ones apply Machine Learning techniques such as Lemmatization, POS, NER, embeddings, semantic similarity, and classification. The article should analyze the text of suicide notes, so it is likely that it also discusses preprocessing and presents performance metrics.” These examples illustrate how easily complex filters can be applied to unstructured information, highlighting the power of these tools to build query systems that do not pose a barrier for the user. In the next chapter, we will show how NLP has been used to build a Telegram bot with capabilities similar to prompts.

### Queries to the selected bibliographic corpus via chatbots

Conversational systems (chatbots) such as ChatGPT or Copilot provide tools and interfaces to enable dialogues between a human and the corresponding AI. The Language Models behind these systems are general-purpose

and, therefore, if we want to customize them with our own documents/information, we must adopt some strategy so the system can acquire that knowledge and respond to our questions.

There are different platforms that offer programming libraries with specialized functions to extract text from your own documents and deliver it to the user through generative AI.<sup>31-33</sup> Your proprietary knowledge is supplied to the large language model, and it can integrate it with its general model to provide corpus-specific answers. These systems are called Retrieval-Augmented Generation (RAG).<sup>34</sup>

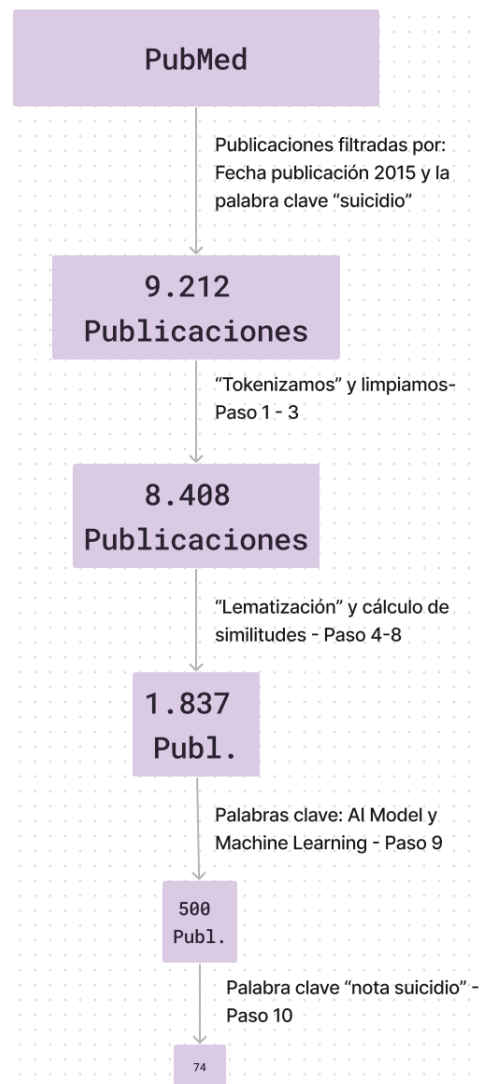


Figure 8. Pipeline or process flow followed and results obtained.

In this section, we show a Telegram chatbot example to create a conversational system in the research domain used throughout this work: youth suicide.

The works selected according to the methodology described form a knowledge corpus on which a researcher can begin their investigation. Consulting these documents is usually done manually to delve into each one's details. Building a question–answer system over that knowledge corpus is efficient and can yield considerable time savings.

Relying again on Large Language Models, we built a conversational system to query any content in the research corpus. The system's ease of use is its main feature, since it relies on interfaces most users are already familiar with. Specifically, a Telegram bot is designed to which the corresponding research questions are posed and, once the documents in the previously selected repository are consulted, the required information is obtained. The next step is to respond to the user in their own language with the findings that answer their questions.

Figure 9 shows an interaction diagram of the conversational system. The basic requirements of this system are:

- Telegram question server.
- Research document database.
- Generative AI server via a large language model for responses.
- Response server via retrieval augmentation.

The Telegram server is installed on a machine that listens for user questions. Anyone can ask questions if they know the bot's name. Questions are stored in a queue pending processing. Another server holds the database of research papers, which have undergone transformations to compute similarities between documents. The most important transformation is to split a concrete document into semantically similar chunks. These chunks are stored in a da-

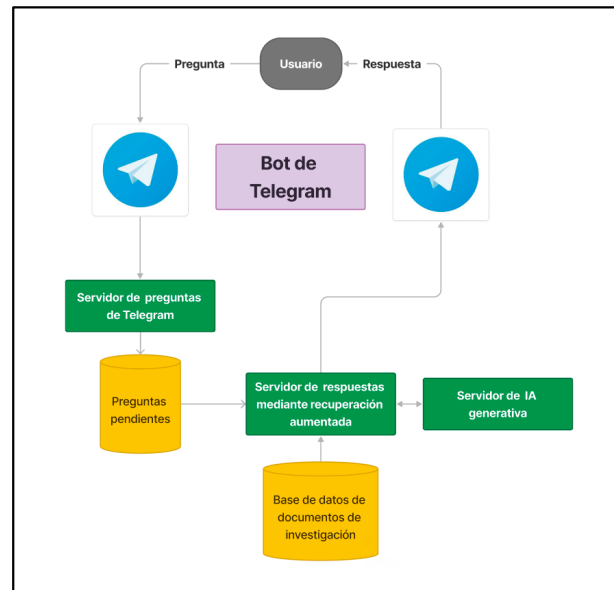


Figure 9. Basic schematic of the Telegram-based system.

tabase in numerical formats thanks to different embedding libraries. When a user asks a question, the bot hands control to the main program, which transforms the question into a numerical format (embeddings). At that moment, it queries the numeric document database, searching for the chunks closest to the question's numerical representation. The query result consists of the  $n$  most similar chunks by semantic similarity. Finally, the text snippets from the retrieved articles are passed to the generative process, so that, based on the LLM, a well-formed response with the required information is generated. This response is sent back to Telegram to be delivered to the user in the appropriate chat. Figure 10 shows the information flow in detail throughout the process.

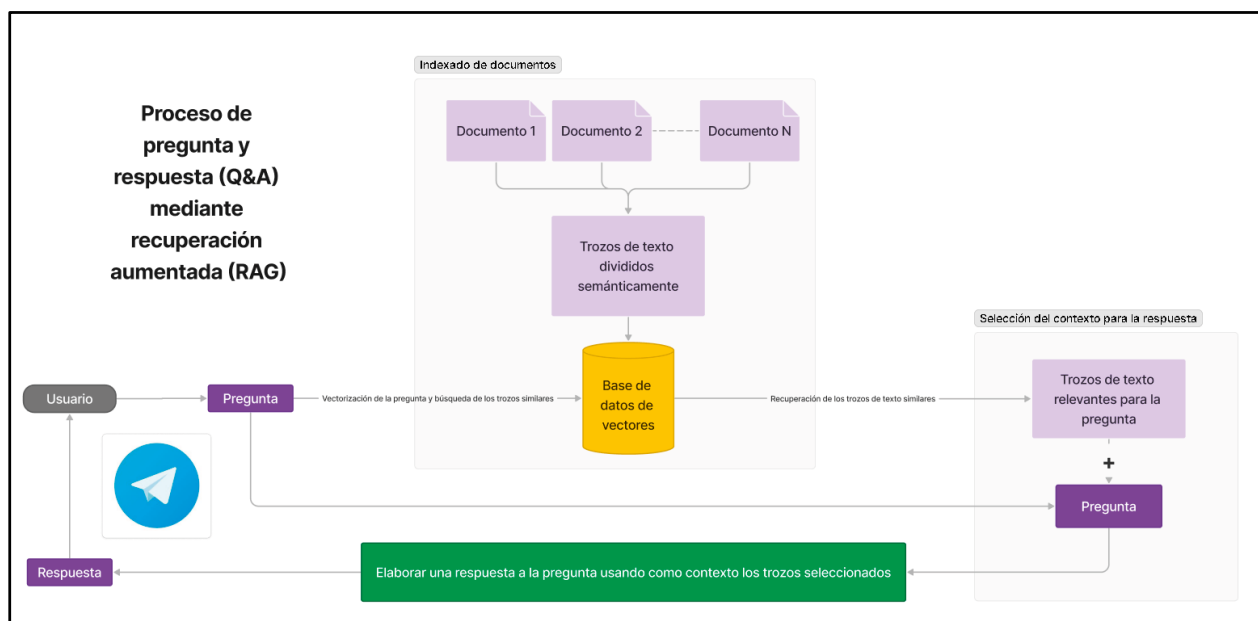


Figura 10. Interaction flow of the question–answering (Q&A) system.

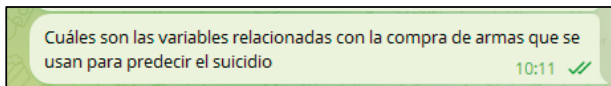


Figure 11. User query example sent to the Telegram bot.

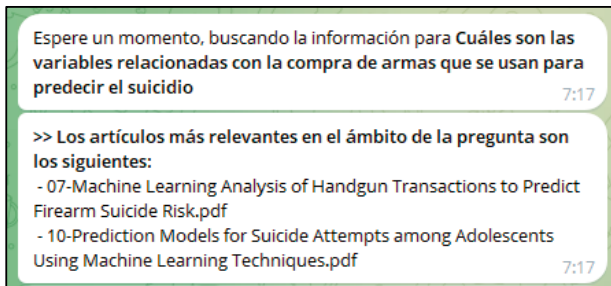


Figure 12. System response listing articles related to the user's question.

Let's see an example of the user experience.

1. We ask the question to the Telegram bot (Figure 11).

2. The bot performs the internal operations and returns the articles that contain content related to the question (Figure 12).

The bot has reviewed the text chunks from these articles related to the question and has passed them to the LLM, which generates the natural-language response shown in Figure 13.

This system allows us to ask colloquial questions about the articles we have selected using the techniques described above. Needless to say, although the system is generalist in that it can serve any scientific domain, the article database will determine the specific domain. Therefore, each researcher or research group could have one or more bots for their different lines of research.

## Conclusions

The proposed methodology significantly reduces the

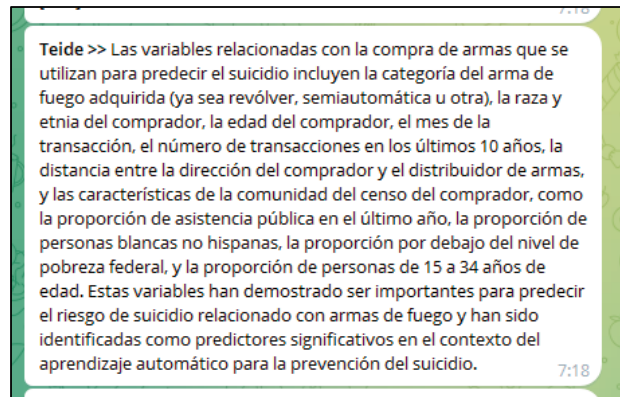


Figure 13. Natural language response generated by the LLM based on retrieved text chunks.

time and effort required for the initial selection of research papers from bibliographic databases.

It is complementary to PRISMA-type methodologies, as it performs the initial search tasks through an automatic selection of bibliography on a specific topic proposed by a researcher.

While classical machine learning and NLP techniques are very effective, they require at least basic knowledge of programming or experience using notebook environments such as Jupyter.

In contrast, LLM-based techniques for searching and selecting research papers are much more accessible to non-programmers, as they allow interaction with a system simply by asking questions via ChatGPT or similar tools.

A customized question-answering system can further improve the efficiency of information searches within a given knowledge corpus.

In summary, AI techniques—particularly those involving NLP and LLMs—promise to be powerful tools that can assist researchers in the search and selection of research papers from bibliographic databases.

## ARTICLE INFORMATION

**Conflict of Interest Disclosures:** None reported.

**Funding:** The authors declare the non-existence of funding in relation to this article.

**Ethical responsibilities:** The authors have confirmed the maintenance of confidentiality and respect for the patient rights, agreement of publication, and transfer of rights to *Revista Española de Urgencias y Emergencias*.

**Article not commissioned by the Editorial Board and with external peer review.**

**Note of the editors:** This is a BOWMAN-generated English translation of the officially indexed Spanish-language article, which should be cited as *Rev Esp Urg Emerg*. 2024;3:184-195. In this translated version, the editors have supervised the process; however, it cannot be ruled out that some errors resulting from the artificial intelligence translation process may have gone unnoticed.

## ACKNOWLEDGMENTS

The authors would like to thank the Cátedra Ca-

jasiete-ULL of Big Data, Open Data and Blockchain, and the Cátedra en Telemática, Robótica y Telecirugía Fundación Canaria Kishoo-Universidad de La Laguna for their support in the development of this work.

## REFERENCES

1. Turing A. Computing Machinery and Intelligence. *Mind*. 1950;236:433-60. doi:10.1093/mind/LIX.236.433
2. Tommasel A, Pablos Sarabia R, Assent I. Re2Dan: Retrieval of Medical Documents for e-Health in Danish. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 2023; 1208-1211. doi: https://doi.org/10.1145/3604915.3610655
3. Carvallo A, Parra D, Lobel H, Soto A. Automatic document screening of medical literature using word and text embeddings in an active learning setting. *Scientometrics*. 2020;125:3047-84. doi: https://doi.org/10.1007/s11192-020-03763-4
4. Peikos G, Alexander D, Pasi G, de Vries AP. Investigating the Impact of Query Representation on Medical Information Retrieval. En: *European Conference on Information Retrieval*, pp. 512-521. Cham: Springer Nature Switzerland, 2023. doi: https://doi.org/10.1007/978-3-031-28238-6\_42
5. Ganguly D, Conrad JG, Ghosh K, Ghosh S, Goyal, et al. Legal IR and NLP: the history, challenges, and state-of-the-art. En: *European Conference on Information Retrieval*, pp. 331-340. Cham: Springer Nature Switzerland; 2023. doi: https://doi.org/10.1007/978-3-031-28241-6\_34
6. Parmar M, Jain N, Jain P, Jayakrishna Sahit P, Pachpande S, Singh S, et al. NLPExplorer: exploring the universe of NLP papers. En: *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal Proceedings, Part II* 42, pp. 476-480. Springer International Publishing; 2020. doi: https://doi.org/10.1007/978-3-030-45442-5\_61
7. Kelly L, Suominen H, Goeuriot L, Neves M, Kanoulas, et al. Overview of the CLEF eHealth evaluation lab 2019. En: *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 10th International Conference of the CLEF Association, CLEF 2019, Lugano, Swit-*

- zerland, Proceedings 10, pp. 322-339. Springer International Publishing; 2019. doi: [https://doi.org/10.1007/978-3-030-28577-7\\_26](https://doi.org/10.1007/978-3-030-28577-7_26)
8. PubMed, National Center for Biotechnology Information (NCBI). (Accessed 12 June 2024). Available at: <https://pubmed.ncbi.nlm.nih.gov/>
  9. Reiter E, Dale R. (1997). Building applied natural language generation systems. *Natural Language Engineering*. 1997;3:57-87. doi: 10.1017/S1351324997001502
  10. Sebastiani F. (2005). Text categorization. In *Encyclopedia of database technologies and applications*, pp. 683-687. IGI Global. doi: 10.4018/978-1-59140-560-3.ch112
  11. Schouten K, Frasinca F. Survey on aspect-level sentiment analysis. *IEEE transactions on knowledge and data engineering*. 2015;28:813-30. doi: 10.1109/TKDE.2015.2485209
  12. El-Kassas WS, Salama CR, Rafea AA, Mohamed HK. Automatic text summarization: A comprehensive survey. *Expert systems with applications*. 2021; 165:113679. doi: <https://doi.org/10.1016/j.eswa.2020.113679>
  13. Grishman R. Information extraction. *IEEE Intelligent Systems*. 2015;30:8-15. doi: <https://doi.org/10.1109/MIS.2015.68>
  14. Caballero, M. A brief survey of question answering systems. *International Journal of Artificial Intelligence & Applications (IJAA)*. 2021;12(5). doi: <https://doi.org/10.1109/MIS.2015.68>
  15. Kusal S, Patil S, Choudrie J, Kotecha K, Mishra S, Abraham A. AI-based conversational agents: A scoping review from technologies to future directions. *IEEE Access*. 2022;10:92337-56. doi: 10.1109/ACCESS.2022.3201144
  16. Agarwal R, Wadhwa M. (2020). Review of state-of-the-art design techniques for chatbots. *SN Computer Science*. 2020;1:46. doi:<https://doi.org/10.1007/s42979-020-00255-3>
  17. Jurafsky D, Martin JH. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (3<sup>rd</sup> ed draft). 2024. (Accessed 17 June 2024). Available at: <https://web.stanford.edu/~jurafsky/slp3/ed-3book.pdf>
  18. Frei J, Kramer F. GERNERMED: An open German medical NER model. *Software Impacts*. 2022;11:100212. doi: <https://doi.org/10.1016/j.simpa.2021.100212>
  19. Padró L, Stanilovsky E. Freeing 3.0: Towards wider multilinguality. In *LREC2012*, 2012.
  20. Paul Mooney, 2019. Medical Symptoms Text and Audio Classification in Kaggle. (Accessed 17 June 2024). Available at: <https://www.kaggle.com/code/paultimothymooney/medical-symptoms-text-and-audio-classification>
  21. Devlin J, Chang MW, Lee K, Toutanova KB. Pre-training of deep bidirectional transformers for language understanding in: *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*. Minneapolis, MN: Association for Computational Linguistics. 2019;4171-86. doi: 10.18653/v1/N19-1423
  22. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language models are few-shot learners. *Advances In Neural Information Processing Systems*. 2020;33:1877-901.
  23. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, et al. Attention is all you need. *Advances In Neural Information Processing Systems*. 2017;30.
  24. BETO: Spanish Bert dccuchile/bert-base-spanish-wwm-cased in Hugging Face. (Accessed 17 June 2024). Available at: <https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased>
  25. Vig J. BertViz, herramienta para la visualización de la atención en modelos BERT. (Accessed 18 June 2024). Available at: <https://github.com/jessevig/bertviz>
  26. Mikolov T, Chen K, Corrado G, Dean J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. doi: <https://doi.org/10.48550/arXiv.1301.3781>
  27. Tatman, R. Pre-trained Word Vectors for Spanish (Dataset Over 1 million 300-dimensional word vectors for Spanish) in Kaggle. (Accessed 14 June 2024). Available at: <https://www.kaggle.com/datasets/rtatman/pretrained-word-vectors-for-spanish?resource=download>
  28. Budanitsky A. Lexical semantic relatedness and its application in natural language processing. technical report CSRG-390, Department of Computer Science, University of Toronto, 1999.
  29. Miller GA. WordNet: a lexical database for English. *Communications of the ACM*. 1995;38:39-41. doi: <https://doi.org/10.1145/219717.219748>
  30. EuroWordNet Project. (Accessed 18 June 2024). Available at: <https://archive.illc.uva.nl/EuroWordNet/data/sampleData.htm>.
  31. Open AI. (Accessed 12 June 2024). Available at: <https://platform.openai.com/>
  32. Meta. (Accessed 12 June 2024). Available at: <https://llama.meta.com/>
  33. Langchain. (Accessed 12 June 2024). Available at: <https://www.langchain.com/>
  34. Lewis PSH, Perez E, Piktus A, Petroni F, Karpukhin V, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*. 2020;vol. abs/2005.11401. [Online]. doi: <https://doi.org/10.48550/arXiv.2005.11401>