# NLP Applied To Portuguese Consumer Law

NUNO CORDEIRO, Instituto Superior Técnico, Portugal
JOÃO DIAS* and PEDRO SANTOS*, INESC-ID, Portugal

As citizens, each and every one of us should know their rights and obligations, especially in a day to day context such as when we pose as a consumer. As of yet, the Portuguese Consumer law is not accessible to the point of being able to insert a sentence written in natural language in a search engine and getting a clear response without first having to scroll through multiple search results. This type of barrier is also what keeps the common citizen from consulting the legislation, especially given the amount of jargon used in legal documents and their structure, which can be difficult to navigate. To solve this issue, we introduce Legal Semantic Search Engine (LeSSE), an information retrieval system that uses a hybrid approach of semantic and syntactic information retrieval techniques, based on the Quin system created by Samarinas et al.

**ACM Reference Format:**
Nuno Cordeiro, João Dias, and Pedro Santos. 2022. NLP Applied To Portuguese Consumer Law. 1, 1 (May 2022), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

The Official Portuguese Gazette (Diário da República) is tasked with the publication of all laws and norms of the Portuguese Republic. Currently, it is exclusively published electronically at DRE[1] by the INCM as a public service that offers universal and free access to all of its content and functionalities. The DRE is composed of a vast set of publications, from which procedures, norms, applications and rules are derived. This online resource currently provides access to all of the Portuguese legislation, as well as services that allow citizens to find the norms and procedures that are inherent to their search.

The current search methodology used in the search engine created for the Portuguese Consumer Law allows a search for legislation that is based on literal keyword search (articles are chosen according to a comparison between the literal keywords in their text and the ones that the user inputs as a search query) which poses some limitations on the accuracy of the results.

With this challenge in mind, our main goal was to engineer a system capable of searching through the Portuguese Consumer Law by providing a query in NL and returning a set of results, in the form of segments of text, with their corresponding information such as the title of the act and its article.

*Both authors contributed equally to this research.
[1]https://dre.pt/dre/home

Authors' addresses: Nuno Cordeiro, Instituto Superior Técnico, Lisbon, Portugal; João Dias; Pedro Santos, INESC-ID, Portugal.

## 2 RELATED WORK

### 2.1 Information Retrieval

Information Retrieval is the procedure in which a system retrieves information from a collection of resources when given a requirement (usually an expression or a query). This task can be applied to numerous domains and is an important aspect of our day-to-day lives.

*2.1.1 Traditional Approaches to Legal Information Retrieval.* Legal Information Retrieval (LIR) is a specific type of Information Retrieval and, therefore, requires different approaches to the way the text is searched. Usually, legal documents are written in a very formal language but search queries written by regular citizens tend to be in a more informal language. This type of imbalance creates a mismatch of vocabulary that damages search if not attended to.

When we look back from LIR into the broad spectrum of Information Retrieval, other options appear. These options do not have a focus on the legal subject but rather on generic text documents. In spite of the particularities of legal documents, such as the connections between different documents, for instance, we can also look at legal articles as text documents — therefore expanding the field of research. When we do this, we are met with several other studied alternatives.

The Okapi BM25 [Robertson et al. 1994], or rather just BM25 is one of those alternatives and it is widely used as an information retrieval ranking algorithm. This algorithm is still used today by search engines to determine the relevance of entries to the searched query, along with TF-IDF (Term frequency - inverse document frequency), on which it relies, as we will see.

BM25 is a bag-of-words retrieval algorithm, which is defined by the representation of text as a set (or *bag*) of words while disregarding their syntax or context. The most popularized version of BM25, introduced in TREC 1994 (a conference on text retrieval) is the following:

Given a query $Q$, containing keywords $q_1, q_2, ..., q_n$, the BM25 score of a document $D$ is defined as:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \quad (1)$$

where $f(q_i, D)$ is the term frequency of $q_i$ in the document $D$, $k_1$ and $b$ are optimization parameters, $|D|$ is the length of the document $D$ in words and $avgdl$ is the average document length in the set of documents. $IDF(q_i)$ is the inverse document frequency of the term $q_i$. It is used as a weight function and it is defined as:

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \quad (2)$$

where $N$ is the cardinality of the set of documents and $n(q_i)$ is the number of documents that carry the term $q_i$.

In a paper [Gain et al. 2019] released for the COLIEE workshop in 2019, this algorithm was used to retrieve legal information based on query search. In the third task (first task of statute law) of the competition, they used the BM25 function to derive the score of each document based on a searched query. It delivered promising results but was still not able to be set apart from other information retrieval techniques — their results were not largely separated.

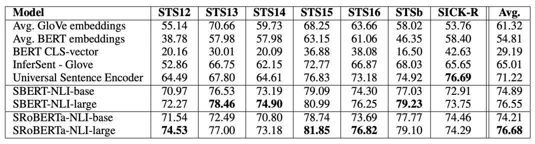
Representation models are one of those alternative information retrieval techniques and their goal is to process information in the documents and queries in order to represent them in a different way. There are three main groups of representation models. The set-theoretic models are the ones that aim to represent documents and queries as sets of words. In this group we find the standard boolean model, which was mentioned previously in the context of legal information retrieval. It is one of the most simple and inexpensive representation models available.

Then we have the algebraic group of models, that represent documents and queries as embeddings with the objective of defining the similarity between documents and queries based on their vectors.

Finally, we have the probabilistic models where the similarities between documents and queries are given by probability of text usage and relevance. In this category we can fit language models, which are a very specific case that has been expanded given their importance in NLP tasks. We will approach systems of this kind in the next subsection.

## 2.2 Natural Language Processing Approaches to Semantic Textual Similarity

*2.2.1 Sentence-BERT.* A team of researchers from Ubiquitous Knowledge Processing Lab (UKP) developed a system, Sentence-BERT [Reimers and Gurevych 2020], heavily based on the BERT model that has averaged promising results in all of the SemEval[2] editions from 2012 to 2016, especially when compared to BERT's averaged embeddings or CLS embeddings — and even when compared to USE and Infersent. (See 1).



| Model | STS12 | STS13 | STS14 | STS15 | STS16 | STSb | SICK-R | Avg. |
|---|---|---|---|---|---|---|---|---|
| Avg. GloVe embeddings | 55.14 | 70.66 | 59.73 | 68.25 | 63.66 | 58.02 | 53.76 | 61.32 |
| Avg. BERT embeddings | 38.78 | 57.98 | 57.98 | 63.15 | 61.06 | 46.35 | 58.40 | 54.81 |
| BERT CLS-vector | 20.16 | 30.01 | 20.09 | 36.88 | 38.08 | 16.50 | 42.63 | 29.19 |
| InferSent - Glove | 52.86 | 66.75 | 62.15 | 72.77 | 66.87 | 68.03 | 65.65 | 65.01 |
| Universal Sentence Encoder | 64.49 | 67.80 | 64.61 | 76.83 | 73.18 | 74.92 | **76.69** | 71.22 |
| SBERT-NLI-base | 70.97 | 76.53 | 73.19 | 79.09 | 74.30 | 77.03 | 72.91 | 74.89 |
| SBERT-NLI-large | 72.27 | **78.46** | **74.90** | 80.99 | 76.25 | **79.23** | 73.75 | 76.55 |
| SRoBERTa-NLI-base | 71.54 | 72.49 | 70.80 | 78.74 | 73.69 | 77.77 | 74.46 | 74.21 |
| SRoBERTa-NLI-large | **74.53** | 77.00 | 73.18 | **81.85** | **76.82** | 79.10 | 74.29 | **76.68** |

Fig. 1. Spearman rank correlation $c$ between the cosine similarity of sentence representations and the gold labels (label that classifies two sentences on the basis of their similarity) for various STS tasks. Performance is reported by convention as $c$ x 100. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset. [Reimers and Gurevych 2020]

Sentence-BERT, or SBERT, is essentially a modified pre-trained BERT model that uses siamese and triplet network structures in order to extract sentence embeddings that are semantically relevant.

The system trains by encoding two sentences in the siamese way, i.e., running two identical networks adjusted with the same parameters on two different inputs — in this case two BERT networks receive each a sentence to encode. These encodings are then passed through a pooling process that the team behind SBERT found to have a better performance with a mean agreggation strategy, as opposed to a max or [CLS] vector strategy. Like we have seen previously, it consists of averaging the token embeddings of a sentence. That way, all of the token embeddings that BERT outputs are joined into one vector, consequently joining a layer of the size of the sequence of tokens into one output. Finally, both sentence encodings are then compared by calculating their cosine similarity. The whole process is depicted in the diagram in 2.

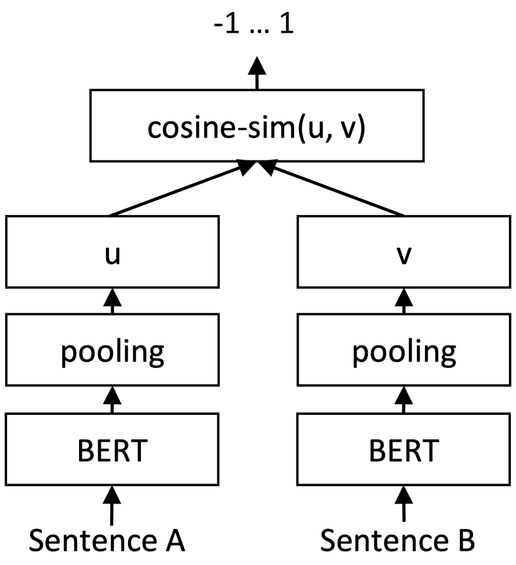


Fig. 2. SBERT architecture for STS tasks. [Reimers and Gurevych 2020]

The training is done using a combination of two datasets — the SNLI [Bowman et al. 2015] and the Multi-Genre NLI [Williams et al. 2018]. The SNLI has a body of 570,000 sentence pairs annotated with the labels contradiction, entailment and neutral. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text.

When developing SBERT, the UKP team compared the performance of different metrics in the evaluation of each sentence pair similarity. They experimented with the most used one, cosine-similarity, but also tried to use negative Manhattan and Euclidean distances. After the experiments, they concluded that the metric used, between the three, wasn't relevant given that the results were roughly the same. Therefore, they continued to use cosine-similarity as the metric of STS.

They also considered using a regression function that maps sentence embeddings to similarity scores but refrained from doing so given the resource exhaustion that would occur.

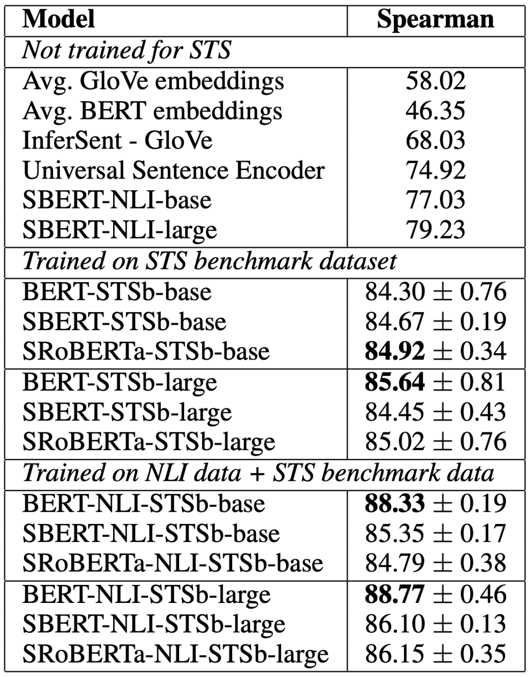
When comparing the performance of SBERT, in STS tasks, two different strategies of training were used — Unsupervised and Supervised Learning. For the unsupervised approach, SBERT only retained the knowledge that it had gained with the pre-training from BERT (based off of Wikipedia) and NLI data. To evaluate this

[2]an international workshop on semantic evaluation (http://alt.qcri.org/semeval2020/)

system, three datasets were used — STS tasks 2012-2016[3], the STS benchmark [Cer et al. 2017] and the SICK-Relatedness [Marelli et al. 2014] datasets. These three datasets include labels for sentence pairs that define, on a scale of 0 to 5, how semantically related they are. SBERT was able to outperform both InferSent and USE on most of the datasets, with the exception of the SICK-R dataset, in which USE gained an edge due to its training on a variety of diverse datasets that seemed to better fit the data in SICK-R. The results can be seen in 1.

For the supervised learning, SBERT was fine-tuned on the training set of the STS benchmark dataset using cosine similarity as the metric for sentence embedding similarity alongside a mean squared error loss function to assess the quality of each prediction. This dataset has proven to be very popular in the evaluation of supervised datasets given the quality of the sentence pairs and its dimensions — it is composed of 8628 sentence pairs that are divided into three categories (*captions*, *news* and *forums*).

Apart from the fine-tuning done only on STSb, another experiment was done by training on the NLI dataset first and then the STSb. This latter option resulted in a considerable improvement. In the paper it was also found that using RoBERTa, a BERT based language model, (instead of BERT) did not make much difference in the final results. These findings are displayed in 3.



| Model | Spearman |
|---|---|
| *Not trained for STS* | |
| Avg. GloVe embeddings | 58.02 |
| Avg. BERT embeddings | 46.35 |
| InferSent - GloVe | 68.03 |
| Universal Sentence Encoder | 74.92 |
| SBERT-NLI-base | 77.03 |
| SBERT-NLI-large | 79.23 |
| *Trained on STS benchmark dataset* | |
| BERT-STSb-base | 84.30 ± 0.76 |
| SBERT-STSb-base | 84.67 ± 0.19 |
| SRoBERTa-STSb-base | **84.92** ± 0.34 |
| BERT-STSb-large | **85.64** ± 0.81 |
| SBERT-STSb-large | 84.45 ± 0.43 |
| SRoBERTa-STSb-large | 85.02 ± 0.76 |
| *Trained on NLI data + STS benchmark data* | |
| BERT-NLI-STSb-base | **88.33** ± 0.19 |
| SBERT-NLI-STSb-base | 85.35 ± 0.17 |
| SRoBERTa-NLI-STSb-base | 84.79 ± 0.38 |
| BERT-NLI-STSb-large | **88.77** ± 0.46 |
| SBERT-NLI-STSb-large | 86.10 ± 0.13 |
| SRoBERTa-NLI-STSb-large | 86.15 ± 0.35 |

Fig. 3. Evaluation on the STS benchmark test set. SBERT was fine-tuned on the STSb dataset, SBERT-NLI was pretrained on the NLI datasets, then fine-tuned on the STSb dataset. [Reimers and Gurevych 2020]

[3]http://alt.qcri.org/semeval2020/

## 2.3 Quin

Quin [Samarinas et al. 2021] is a fact-checking system that was developed during the outbreak of COVID-19 with the purpose of providing the public with an automated fact-checking system capable of examining the veracity of claims surrounding the topic of COVID-19. It was later repurposed as a general fact-checking system, capable of verifying open-domain claims.

Quin works in three stages (see 4) — in the first, the query goes through a BM25 sparse retriever, from which the top scoring 500 results are extracted. In the second stage, parallel to the first, the query is encoded using QR-BERT, a BERT model specifically designed to work in the context of question answering, trained with a large dataset constructed using NLI. After the query is encoded, it goes through a Faiss index to search for the passages that best resemble the query in semantic value, using a cosine similarity function to compare between embeddings, and the best 500 results are extracted from this stage.

The union set of the results from stages 1 and 2 are then used in the third and final stage, where these go through a relevance classifier, that is essentially a BERT model fine-tuned on a large dataset of query-passage pairs that applies a linear transformation to the embedding of the [CLS] token in order to retrieve a score from each query-result pair in the union set. These results are then reordered according to the score attributed by the relevance classifier, and this is the final order used to display the results in the search engine.

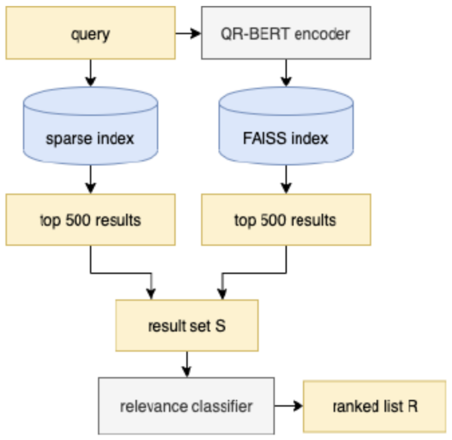


Fig. 4. Diagram of the architecture of Quin. [Samarinas et al. 2021]

This system is very relevant to our problem, given that the motivation is somewhat analogous to ours, albeit in a different context, and it combines traditional information retrieval with a NLP-based approach that is capable of assisting a traditional keyword search with a semantic component, hence the reason we decided to base our system off of this approach.

## 3 LEGAL SEMANTIC SEARCH ENGINE

The goal of this thesis is to introduce a system that merges a common document retrieval technique with semantic search abilities on the Portuguese consumer law. In this chapter, we will start by looking at

LeSSE, a search engine we specifically created to answer questions on the topic of Portuguese consumer law that uses state-of-the-art search techniques, and is based on Quin [Samarinas et al. 2021].

This system was developed for this thesis in the context of the project *Descodificar a Legislação*, a research collaboration between INESC-ID and INCM. The goal of this project was to make the Consumer Law more accessible and understandable by the Portuguese citizens. This was done by combining popular document retrieval techniques with the recent advancement of Machine Learning and NLP to provide semantic search capabilities to the search system.

This chapter starts with an overview description of the system, that will broadly explain the pipeline step by step. Then, we will address each system component in a more detailed way, as well as the training process for the language model used in some of the components. Ultimately, we have a description of how the results are selected, organized and presented to the user.

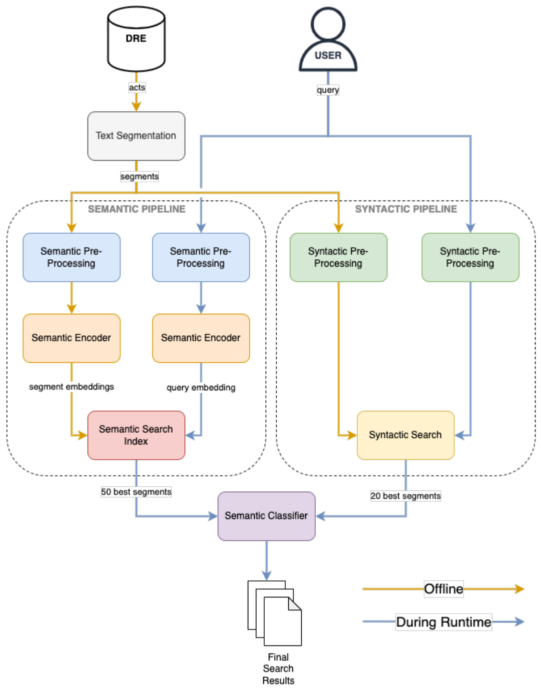


Fig. 5. LeSSE

## 3.1 System Overview

Our system takes a hybrid approach, in which it combines a traditional (syntactic) information retrieval algorithm with a semantic search. The syntactic search allows the users to search for literal terms, such as names or titles included in the legislation, and the semantic search assists in case the answer contains juridical jargon that the user did not use in the query. The semantic search is able to identify synonymous words and expressions that the user may be interested in. A visual description of the system architecture can be found in Figure 5.

First, the system starts by pre-processing all of the law documents (acts) before any search is to be initiated. This will save time and resource exhaustion since all of the acts will be used in each search, therefore avoiding repetitive computations, given that they are always stored in the database. The search query will also undergo pre-processing, but since the queries will be inevitably different each time a search is performed, it is done during search time.

The pre-processing is the same for the acts and the query with the exception of one step — the acts go through text segmentation first, in which the text of each act is separated into segments, which are pieces of text separated by line breaks, and each one of them contains information about its location in the document (e.g. chapter 3, section 2, article 24) so that they can be later referenced in the results.

After that, both the segments and the query go through Semantic and Syntactic Pre-Processing — two distinct text processing steps that are needed to prepare the segments and the query for the Semantic Extraction and Syntactic Similarity, respectively. It is worth mentioning that the pre-processing is done at different stages for the segments and the query. The segments are pre-processed when the system is started and the query is pre-processed during runtime, right after it is inserted into the search engine.

After the semantic pre-processing is complete, we move on to the semantic extraction, which is the step in which the segments and the query are encoded into embeddings.

With the segment embeddings that were generated, a search index is created and stored locally in the server that hosts the search service. This allows the similarity search to be more efficient, and given the persistent nature of the corpus, the index won't need to suffer changes as long as the corpus does not change.

Additionally, before the system is up and running, the segments are used to create a structure that holds each word present in the segments (every word in the corpus), that is used by the syntactic similarity algorithm (BM25) to determine a syntactic similarity score for each pair of segment-query.

Once all of this is complete, the system is then ready to receive a query. In search time, the query goes through the syntactic pre-processing and is then used to calculate a similarity score for each segment. The 20 segments with the highest scores are selected. Consecutively, the query also goes through the semantic pre-processing, before being converted into an embedding, that is afterwards used to perform a search in the index that was created with the segments. The search index returns the scores of semantic similarity between each segment and the query, based on the similarity of their corresponding embeddings, and, at that point, the 50 segments with the best scores are selected.

In the final stage, the 50 best segments from the semantic search index and the 20 best segments from the syntactic similarity search are then united into a single set, but since the scores from the syntactic and semantic searches are on different scales and are not comparable, the results are then passed on to a trained semantic similarity model that assigns each result pair with a score that signifies its similarity to the initial query. Ultimately, the results are ordered according to these scores and returned to the user.

## 3.2 Semantic Pipeline

*3.2.1 Semantic Pre-Processing.* As opposed to the pre-processing done for the syntactic search, the one that is used for the semantic search does not involve any of the steps mentioned prior, but instead rests on the Bert Tokenizer from the Hugging Face[4] library.

*3.2.2 Semantic Encoder.* After the semantic pre-processing, we are left with segments (and a query) that are prepared to be received by the Semantic Encoder, which will then generate segment embeddings off of those segments.

Semantic Encoder is, in its core, BERTimbau [Souza et al. 2020], a BERT model trained on the BrWaC [Wagner Filho et al. 2019], a large corpus that was constructed using the Brazilian Portuguese Web as a source. For the purpose of simplifying the fine-tuning and evaluation, we have used BERTimbau Base (BERT-Base) with hidden size 768.

BERT-Large, with hidden size 1024, is known to present better results than BERT-Base, but since fine-tuning it requires more computational power, BERT-Base came as the best choice in terms of time, performance and computational limitations.

In this step, the ultimate goal is to create a segment embedding, but since BERT only generates token embeddings we will be using the average embedding strategy to generate a segment embedding out of all the tokens in it. Therefore, once BERTimbau finishes embedding the tokens in the segment, all of the token embeddings are then averaged into a single segment embedding array.

Once this has been done, the embedding array is then normalized. This is a requirement for the Faiss [Johnson et al. 2019] index that we are using and it also facilitates the selection of a fixed threshold for the maximum cosine distance between arrays, since these distances will then be comprised between 0 and 1 after the arrays are normalized.

For the purpose of adjusting the Semantic Encoder to the vocabulary used in the law documents and the queries (European Portuguese, formal language in the law documents and informal — sometimes formal — in the queries), we had to fine-tune the model in order to teach it to recognize popular semantic pairs between queries and law segments. To do this we used the Trainer[5] class from the Hugging Face library, providing a training and evaluation loop for PyTorch, optimized for Hugging Face Transformers classes.

To train the Semantic Encoder we used the Manual Annotations dataset on a task of sequence classification. The model was trained for 1 epoch, at an initial learning rate of 7.40546e-05, weight decay 0.244911, with a training batch size of 32 per GPU (a total of 64, considering the training was done on two GPUs).

Since we are using the Base version of BERTimbau in the Semantic Encoder, with hidden size 768, the generated segment embeddings are feature vectors with 768 dimensions (features).

After the creation of all of the segment embeddings, they are then stored in a Faiss search index for future use (every time a search query is received).

The reason why we opted for BERTimbau as opposed to M-BERT is because of the way these models were trained. M-BERT was trained for multiple languages, including Portuguese, but in a much more modest way compared to other languages since the corpus is proportional to the available source material in each language. And since M-BERT needed to be trained in various languages, the trade-off was between number of languages included and the size of the training corpus for each language. Given that M-BERT was specifically designed to be pre-trained in multiple languages, its performance cannot be compared to the one from a model that was specifically trained in one language solely.

*3.2.3 Semantic Search Index.* After every segment in the corpus is encoded, they are added to a Faiss Search Index. The Index being used is the IndexFlatIP — providing an exact search for inner product. Since there is no index that provides an exact search based on the cosine similarity of the arrays, we chose the inner product considering that the cosine similarity is simply the inner product between normalized vectors — and that is why all of the embeddings are normalized before entering the index, in the final stage of the Semantic Encoder.

The Faiss library possesses two functions that easily allow us to convert the index in memory to a binary file when it is created (write_index) and read that binary file and bring the index back to memory upon the initialization of the search system (read_index). This is convenient given that the creation of the index is a task that takes quite a few minutes (20-30) to complete, and would be a waste of time to repeat it every time the system initializes.

After the query is encoded into an embedding, it is then used to search for the 50 arrays that are closest to it (in cosine similarity). We then use a threshold of 0.5 to eliminate any array that had a score below significant — these arrays represent the segments that are not relevant to the query that was searched. The remaining arrays — the ones with a score above 0.5 — are the ones that represent segments that are relevant to the query, and these segments will go to the latter stage in the system — the Semantic Classifier.

## 3.3 Syntactic Pipeline

*3.3.1 Syntactic Pre-Processing.* Before forwarding the query to the syntactic similarity search, the query needs to be processed in order for it to be recognized even when it is not written in the same way that it exists in the dictionary, due to various concerns.

The syntactic similarity search that is used in the system is the BM25 algorithm, which uses a bag-of-words strategy. Essentially, it means we need to divide a phrase into words. The words belonging to the query are therefore compared with the words in the dictionary (corpus/segments) and, for two words to be considered the same, they must share every Unicode[6] character in the same order.

In the syntactic similarity search, the words of each segment and query are used to calculate a similarity score. For two words to be considered the same they have to be be exactly the same — that is, they must share every unicode character in the same order. When two words are compared by the algorithm, one starting with upper case, and the other with lower case, they are deemed unequal.

Because of this constraint, the syntactic search must follow a syntactic pre-processing that consists of five main steps:

[4]https://huggingface.co
[5]https://huggingface.co/transformers/main_classes/trainer.html#transformers.Trainer
[6]https://home.unicode.org

**NLTK Word Tokenization:** Using the NLTK word tokenization tool for the Portuguese language, each segment and query is separated into tokens, which are words defined in the NLTK Portuguese dictionary. This helps us construct the bag-of-words necessary for the BM25 algorithm, and is especially helpful in recognizing a word that is separated by a hyphen, for instance, which would have been viewed as two single words by a simple space-separator.

**Removal of Punctuation:** Since the previous step does not remove punctuation, every punctuation character is removed so that it does not count as a word. Otherwise, the segments and the query could be compared using, for instance, the number of commas that they contain, which is not our goal.

**Word Lowering:** Words that contain uppercase letters are lowercased so they can be identified as being the same word. This ensures that when the user submits a search query that mistakenly includes an uppercase letter in the wrong place, the words in the query can be identified as belonging to the corpus.

**Stop-Word Removal:** In order to perform a relevant keyword search on the corpus, the segments of the corpus and the query need to go through a process of stop-word removal since, by definition, they offer no additional meaning to the sentences. To do this, we use the NLTK Portuguese stop-word dictionary to remove all of the irrelevant words. In this corpus of stop-words, there exist words such as: *a*, *ao*, *aos*, *aquela*, *aquelas*, among many others. The process of removal is quite simple and it involves keeping only the words in the segments/query that are not present in the corpus of stop-words. This a major step in the text processing since words such as *de* appear a lot in the Portuguese language, and the similarity between two phrases (a segment and the query) should not be judged by their amount of *de*s, which provide no contextual information about the phrase.

**Unidecode:** This last step ensures that every word in the segments and the query do not have any special characters such as *é*, *à*, *ç*, among others. It also ensures that the algorithm is able to recognize every word that the user typed with a missing accent. Every word goes through the function unidecode from the Unidecode[7] library.

*3.3.2 Syntactic Search.* Following the Syntactic Pre-Processing, the processed segments and the query will go to the Syntactic Search, where the query will be compared to the segments, on a syntactic level. For this purpose, we perform a BM25 text search using the BM25Okapi[8] class from the rank_bm25 python library but, before any search is done, at the initialization stage of the search system, the BM25Okapi object is initialized using the corpus, i.e., all of the segments in the corpus that have been pre-processed. Since it only takes a few seconds, there is no need to store the binary object in memory and it is done during loading time, every time the system loads.

[7]https://github.com/avian2/unidecode
[8]https://pypi.org/project/rank-bm25/

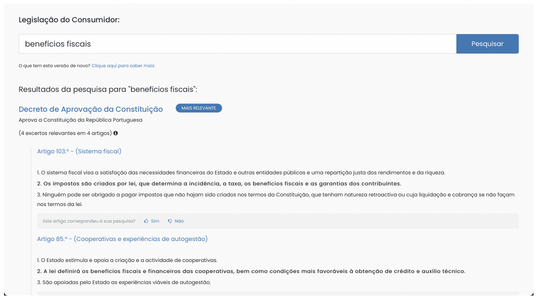
*3.3.3 Semantic Classifier.* At this stage, the segments that were chosen from the Semantic and the Syntactic Searches are collected into a set of segments that will be reordered by the Semantic Classifier, which is none less than a BERTimbau model trained in the same way that the one in Semantic Encoder is fine-tuned. However, their purposes are separate — the Semantic Encoder uses a base BertModel class to generate the embedding for each sentence and then they are compared in the search index based on their cosine similarity whereas the Semantic Classifier uses a BertForSequenceClassification class and the semantic similarity score is calculated by applying a softmax function to the logits that are returned by the model, upon receiving both sentences (segment and query) as input.

The Encoder + Search Index combination is used at a stage where comparison speed needs to be high, since we need to compare the query to every single segment in the corpus. At a later stage, in the Semantic Classifier where, at most, we have 70 segments, we are allowed to use a BertForSequenceClassification model to compare the query to the final segments and classify them in the order of relevance.

## 3.4 Results Selection and Presentation

The final results are shown in the following manner: The acts are ordered according to the sum of the scores of its segments, in a descending order. Thus, the act with the highest sum of its segments' scores is at the top of the results list. In each act, the ordering of the articles follows the same strategy — the articles in each act are ordered according to the sum of its segments' scores.

In Figure 6 we have an example of the results interface where we searched for the query terms "benefícios fiscais" and it returned an act in first place named Decreto de Aprovação da Constituição, and inside it two articles, Artigo 103º and Artigo 85º.



O que tem esta versão de novo? Clique aqui para saber mais

Resultados da pesquisa para "benefícios fiscais":

Decreto de Aprovação da Constituição

Aprova a Constituição da República Portuguesa

(4 excertos relevantes em 4 artigos)

Artigo 103.º - (Sistema fiscal)

1. O sistema fiscal visa a satisfação das necessidades financeiras do Estado e outras entidades públicas e uma repartição justa dos rendimentos e da riqueza.
2. **Os impostos são criados por lei, que determina a incidência, a taxa, os benefícios fiscais e as garantias dos contribuintes.**
3. Ninguém pode ser obrigado a pagar impostos que não hajam sido criados nos termos da Constituição, que tenham natureza retroactiva ou cuja liquidação e cobrança se não façam nos termos da lei.

Este artigo correspondeu à sua pesquisa?

Artigo 85.º - (Cooperativas e experiências de autogestão)

1. O Estado estimula e apoia a criação e a actividade de cooperativas.
2. **A lei definirá os benefícios fiscais e financeiros das cooperativas, bem como condições mais favoráveis à obtenção de crédito e auxílio técnico.**
3. São apoiadas pelo Estado as experiências viáveis de autogestão.

Fig. 6. Search Results Example

## 3.5 Model Training

As previously stated, the language model that was used for both the Semantic Classifier and Semantic Encoder was a BERTimbau base model fine-tuned on the annotations dataset. For this fine-tuning task, we were able to use a machine with 2 NVIDIA GeForce RTX 3090 GPUs, each with 24GB of memory and 10496 cores.

Prior to training, we ran a hyper-parameter optimization in order to optimize the learning capabilities of the model and its performance on unseen data. We did this by using a function in the Trainer class from the HuggingFace python library, which we also used to train the model. This class provides an API for feature-complete training in PyTorch, which simplifies much of the process and abstracts the researcher from the training loop, focusing on the optimization process. The API also supports distributed training on multiple GPUs/TPUs, which was beneficial to us since we used 2 GPUs instead of just one, and we would like them to be used in parallel to guarantee that the training process finishes faster.

*3.5.1 Hyperparameter Optimization Techniques.* In practice, hyperparameter optimization is nothing but a search for the set of parameters that optimize the training process and maximize (or minimize) the validation metrics, which in our case are the validation accuracy and loss. Usually, hyperparameter optimization can be done using various techniques, but the three most commonly used ones are Hand Tuning, Random (or Grid) search and Bayesian search.

These three techniques, however, possess two downsides. The first downside is the time it takes to find the hyperparameters that optimize the performance of the model. With Hand Tuning, the researcher must try a set of hyper parameters by training the model with that set and then evaluate the performance of the model. The researcher must keep this cycle until they are satisfied with the performance of the model. This task can take a lot of time, and can sometimes even take weeks or months to reach the perfect set of hyper parameters.

Bayesian Optimization is a technique that automates the hand tuning process by using the Bayes theorem to make changes to the hyper parameters at the end of each training run, which makes it a sequential training optimization, and therefore it can also become a very slow process.

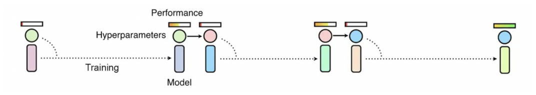


Fig. 7. Bayesian Optimization

In Grid Search, a set of neural networks are trained independently in parallel and, at the end, the hyperparameters from the model with the highest performance are selected. The number of different combinations of hyperparameters can go around the dozens or even hundreds, which means that dozens or hundreds of models will be trained and only a small fraction of those will have a high performance. This, in turn, means that most of the models that were trained will not be used and, therefore, a high amount of computer resources were wasted training them — making this the second downside to using one of the most commonly used optimization techniques.

PBT [Jaderberg et al. 2017] is an optimization technique that combines Grid search and Hand Tuning. PBT starts in the same way of Grid Search — by training a set of models with random hyperparameter values — but instead of training the set of models independently, it allows the models to share information with each other and reconfigure their training hyperparameters according to the most promising models.

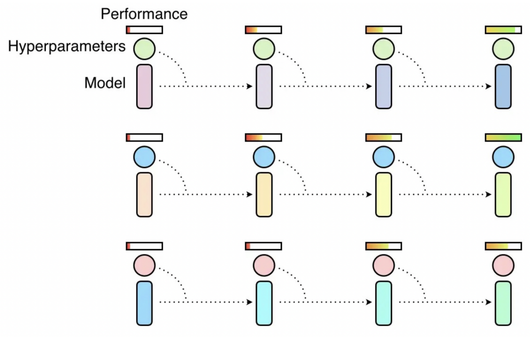


Fig. 8. Grid Search

Given these advantages, we decided to optimize the hyperparameters by using a PBT algorithm that was available as a scheduler option with the same name, in the Ray Tune library, that is integrated into the Trainer class function hyperparameter_search.

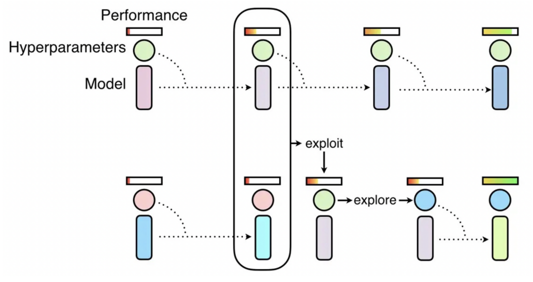


Fig. 9. PBT

*3.5.2 Model Optimization and Training.* In order to find out which type of data would train the model in the best way possible for this task, we studied the use of 4 datasets for the Consumer Law Corpus — Manual Annotations, ICT, Semantic Pairs and Manual Annotations + Semantic Pairs — and for the Retirement Corpus we used 2 datasets, Manual Annotations and Semantic Pairs.

To choose which training dataset would return the best results we followed the same evaluation procedure for all of them. We first divided the datasets into 3: The training set, used to train the model; the validation set, that was used to evaluate the performance of the model during training; and the test set, which was used to evaluate the performance of the model on unseen data (different from the validation set), after the training is complete.

The Manual Annotations dataset was divided into 2 subsets — training and validation/test. Since we did not have many golden labels, we decided to split the dataset cautiously, prioritizing the validation/test set given that these sets had to have manually annotated queries, in order to check the real performance of the system. The validation and test sets are the same one given the same reason (not enough manual annotations). Instead of using percentages to split the dataset, we used the number of queries in the dataset — 100 for the validation/test sets, and the remaining (221) for the training set. We chose this query-split approach as opposed to a traditional percentage-split approach, given that, if we had chosen to divide to set according to traditional percentages, such as 80-10-10 (training-validation-test), we would have only had 32 queries to test the performance of the system, which is just not enough, given that there are plenty of topics in the consumer law domain and, with 32 queries, the probability of covering all of those topics is very low.

The other datasets (ICT and Semantic Pairs) were also divided into 2 subsets — training and validation — but in this case following percentages, 80% and 20%, respectively. The reason why we didn't divide these datasets into 3 was because we only needed one test set to test all the models, and we chose the annotations dataset because that was the most reliable since it included manual annotations created by experts, therefore the veracity of those annotations is guaranteed, as opposed to the ones automatically generated, which might not be correct.

Before the division, every dataset is shuffled in order to guarantee that every training batch is representative of the dataset, as a whole. If we did not shuffle the data, we would risk creating batches that have similar data, which would set off the gradient estimate and, therefore, lead the training of the model in the wrong direction.

Then, we ran a hyperparameter optimization for each dataset separately. This step was necessary due to the scale and diversity of data present in each dataset. A selection of parameters and their model performance are displayed in Table 1. The final training hyperparameters were chosen according to the accuracy, while monitoring the loss value which is calculated with a Cross Entropy loss function. This function is defined as:

$$loss = -w_y \cdot \log \frac{\exp(x_y)}{\sum_{c=1}^{C} \exp(x_c)} \cdot y \tag{3}$$

where $x$ is the input, $y$ is the target, $w$ is the weight and $C$ is the number of classes, which in our case is 2 (0 — not relevant, and 1 — relevant).

Table 1. Example of Training Hyperparameters for Manual Annotations of the Consumer Law Corpus

| Weight Decay | Learning Rate | Training Batch Size Per GPU | Epochs | Validation Accuracy | Validation Loss |
|---|---|---|---|---|---|
| 0.261884 | 2.42913e-05 | 32 | 4 | 0.837746 | 0.934985 |
| 0.244911 | 7.40546e-05 | 32 | 1 | 0.843621 | 0.634938 |
| 0.204092 | 9.25682e-05 | 16 | 1 | 0.84328 | 0.638671 |
| 0.261884 | 1.61942e-05 | 32 | 4 | 0.83131 | 1.09068 |
| 0.0596527 | 9.41399e-05 | 16 | 1 | 0.818426 | 0.500694 |

## 4 PERFORMANCE OF LESSE IN THE PORTUGUESE CONSUMER LAW

The primary purpose of the evaluation was to compare the performance of LeSSE in the Portuguese Consumer Law corpus to the baseline, BM25. After choosing the best model from each training session with the different datasets, we compared their performance on the same test set. In the following table, we present the accuracy results for the different combinations of search algorithms and training datasets used. The accuracy is divided into 4 categories (TOP 1, 3, 5 and 12) — each TOP $x$ category represents the percentage of test queries that the system got right in the first $x$ results. And so, for instance, this means that the Baseline system was able to fetch at least one of the correct results, in the first 3 search results, in 70.0% of the test queries (TOP 3/Baseline). The accuracy is measured by comparing the act in the result with the act in the golden label.

The third column shows the results obtained with the LeSSE system without fine-tuning (training) the model. Despite the fact that these results are not the best, this option manages to achieve a better result than the baseline. The fourth column shows the performance of LeSSE when the models were trained using the Manual Annotations dataset, and these are the best results achieved, especially when comparing with the baseline.

Table 2. LeSSE Accuracy in the Portuguese Consumer Law

| Accuracy (%) | | | |
|---|---|---|---|
| Results Measure | Baseline (BM25) | LeSSE | |
| | | No Training | Trained with Manual Annotations |
| TOP 1 | 42.0 | 44.0 | 55.0 |
| TOP 3 | 70.0 | 74.0 | 89.0 |
| TOP 5 | 71.0 | 88.0 | 96.0 |
| TOP 12 | 75.0 | 95.0 | 99.0 |

The reason why all the accuracy percentages in Table 2 are whole numbers is because the test set from the consumer law corpus had 100 query-answer pairs. So, for instance, this would mean that the baseline search algorithm (BM25) got the first result right in 42 out of the 100 test queries (TOP 1, Baseline (BM25)).

After comparing the different iterations of the system for the Consumer Law context, we chose the best configuration based on the TOP 3 measure, which was deemed more relevant in the search task.

## 5 PERFORMANCE OF LESSE IN THE ABSENCE OF MANUAL ANNOTATIONS

The second evaluation sought to answer whether the LeSSE would be ready to answer questions correctly when there would be no manual annotations or jurists to annotate them. This was a relevant question to pose, since there may come a scenario where there are no manual annotations available for a specific domain in the Portuguese Law, as it is the case now for any domain other than the ones we have worked with — Consumer Law and Retirement Law. To prevent this issue, we studied the option of using automatically generated annotations, by following two generation techniques — ICT and Semantic Pairs generation. Additionally, we also tried a mixed configuration of Manual Annotations and Automatic Annotations (Semantic Pairs) to see whether it improved upon the Manual Annotations configuration. And so, upon training LeSSE with these automatically generated datasets, we compared their performance against the Baseline.

Table 3. Testing Accuracy with Automatic Annotation Datasets

| Accuracy (%) | | | |
|---|---|---|---|
| Results Measure | LeSSE | | |
| | Trained with Automatic Annotations | | |
| | ICT Dataset | Semantic Pairs | Manual Annotations + Semantic Pairs |
| TOP 1 | 45.0 | 51.0 | 50.0 |
| TOP 3 | 76.0 | 77.0 | 88.0 |
| TOP 5 | 84.0 | 90.0 | 96.0 |
| TOP 12 | 95.0 | 95.0 | 98.0 |

In Table 3, the fifth column presents the accuracy results of LeSSE when trained with the ICT dataset. Despite not being the best in the bunch, it managed to surpass LeSSE when no fine-tuning was done (when comparing TOP3), and it also showed a significant improvement over the baseline. This means that it could be a good option when there are no manual annotations to fine-tune on, since it is relatively fast and inexpensive to generate.

The sixth column contemplates the results from training with Semantic Pairs, automatically generated from the corpus. When comparing with the manual annotations dataset in the TOP3 category, training with this dataset did not improve the performance of the model, but it did perform better than LeSSE when trained with the ICT dataset. It also showed to be far superior the baseline, and even scored higher than LeSSE with no training, which was indicative that it could be useful in a scenario where there are no annotations. The generation of this dataset was the product of a master thesis from another student, and it required the help from two linguists that annotated the segments with syntactic and semantic functions of the words and expressions, so its generation was slower and more expensive than the ICT one.

Mixing the manual annotations with the semantic pairs did not show an improvement over the manual annotations, and we concluded that it had to do with the incongruousness of the data — the manual annotations were quite different from the automatic ones in terms of topics covered, but also in format (the manual ones were more naturally written than the automatic ones), and that created an inconsistent dataset which was not as fit for training as the manual annotations themselves. Another issue with this mixing strategy comes from the fact that the datasets are not balanced in terms of quantity — the automatic annotations far surpassed the manual ones, since those were easier to generate, and that created an imbalance. However, it still managed to score higher than all the other configurations in the TOP3 category (apart from manual annotations), so it could prove to be quite useful when there are not enough manual annotations to cover all of the topics in the domain, or the quantity of annotations is just not enough.

## 6 PERFORMANCE OF LESSE IN A DIFFERENT LAW DOMAIN

In addition to the Consumer Law domain, another challenge was defined in the *Descodificar a Legislação* project, which was to make the developed system work with another law corpus (another law domain). This task functioned as a test to see how well LeSSE would work when dealing with another domain or subdomain of the Portuguese Law. For this purpose, we were provided with a few annotations on the domain of Retirement Law (Estatuto da Aposentação) and applied the same methodology that we had done for the Consumer Law corpus. The annotation dataset consisted of 111 queries pointing to 298 segments. This dataset was split similarly to the one used in the consumer law domain, by using a set number of queries for the validation and test sets (the same set for both, due to lack of annotations) and the remaining queries were used for training. In this case, we used 32 queries for the validation and test sets, and the remaining 79 for training.

In Table 4, we are presented with the results of the experiment, homologous to the one done for the Consumer Law domain, with the exception of the way in which the accuracy is measured. In the consumer law domain, the accuracy is measured by comparing the act in the result with the act in the golden label, but in this domain we only have 2 acts, which would automatically mean that there is a 50% chance that a random answering search system would get the right result. Since there is no relevance to that kind of metric, we decided to measure the accuracy on the article level — we compared the article in the result with the one on the golden label — and this led to a more fair comparison between domain performances.

Table 4. LeSSE Accuracy in the Retirement Law

| Accuracy (%) | | | | |
|---|---|---|---|---|
| Results Measure | Baseline (BM25) | LeSSE | | |
| | | No Training | Trained with Manual Annotations | Trained with Automatic Annotations (Semantic Pairs) |
| TOP 1 | 43.8 | 43.8 | 46.9 | 40.6 |
| TOP 3 | 50.0 | 68.8 | 78.1 | 71.9 |
| TOP 5 | 50.0 | 75.0 | 90.6 | 84.4 |
| TOP 12 | 50.0 | 75.0 | 90.6 | 84.4 |

The results obtained from this experiment were much the same — that is, the manual annotations were the best dataset to train on, since it far surpassed the others in any category of accuracy and the automatic annotations (in this case Semantic Pairs were used) proved to be the second best option, with the exception of the TOP1 category, in which it did not score as high as the others, but it was compensated by the other accuracy scores.

## 7 CONCLUSION

The proposal of this project was to create a search system that would connect the Portuguese citizens to their Consumer Law by modifying how the search was made. The current Portuguese Consumer Law search engine works by searching keywords in the articles. Our improved version combines a more refined keyword search with a semantic search. For the keyword search we used a well established algorithm called BM25, in its original version, and for the semantic search we used a BERT language model, in a tailored pipeline.

In order to produce the desired results, the language model had to be trained on a corpus that included legislative jargon. This was an important step to ensure that the model would be able to create the right relations between similar words that it had not seen in the pre-training stage (first training with an extensive corpus). It was also important since the pre-training was done using a Brazilian Portuguese corpus, and because we needed it to be able to recognize the European Portuguese vocabulary, it had to be fine-tuned on an European Portuguese corpus.

The training corpus was constructed with the help of jurists from INCM, who annotated questions extracted from the DRE search database with passages from the Portuguese Consumer Law. This allowed us to pair questions with segments (smallest fragment of text in this context) from those passages and those were used, not only to train the model, but also to evaluate its performance during the training stage and to test after it had been trained.

We also tried training the model on other corpora to see how they would influence the performance and how the search engine scaled by training with automatically generated annotations. We observed that LeSSE is quite capable of achieving a high performance when trained with manual annotations (89.0% TOP3 accuracy), especially when compared to the Baseline (70.0%). We also observed that, in the absence of manual annotations the system was able to perform quite well, scoring 76.0% and 77.0%, when trained with an ICT dataset and a Semantic Pairs dataset, respectively.

Ultimately, we also tested the performance of LeSSE in a different law domain. We used a few annotations from the Retirement Law Corpus, which is a smaller corpus than the one originally used (Consumer Law), but, despite having less annotations to train and test on, the experiment proved that the system is fit to work on any domain of the Portuguese law, scoring 78.1% TOP3 accuracy and a decent 71.9% when trained with automatic annotations (in lack of manual ones).

## ACKNOWLEDGMENTS

This work was funded by a partnership between INESC-ID and INCM, in the context of my masters thesis.

## REFERENCES

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/d15-1075 arXiv:1508.05326

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, 1–14. https://doi.org/10.18653/v1/S17-2001

Baban Gain, Dibyanayan Bandyopadhyay, Tanik Saikh, and Asif Ekbal. 2019. IITP in COLIEE@ ICAIL 2019: Legal Information Retrieval using BM25 and BERT. *Competition on Legal Information Extraction/Entailment 2019* (2019).

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*.

Nils Reimers and Iryna Gurevych. 2020. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. https://doi.org/10.18653/v1/d19-1410 arXiv:1908.10084

S E Robertson, S Walker, K Sparck Jones, and M M Hancock-Beaulieu. 1994. Okapi at TREC-3. *Proceedings of the Third Text REtrieval Conference* (1994).

Chris Samarinas, Wynne Hsu, and Mong Li Lee. 2021. Improving Evidence Retrieval for Automated Explainable Fact-Checking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*. 84–91.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020. BERTimbau: Pretrained BERT Models for Brazilian Portuguese. *Intelligent Systems Lecture Notes in Computer Science* (2020), 403–417. https://doi.org/10.1007/978-3-030-61377-8_28

Jorge A. Wagner Filho, Rodrigo Wilkens, Marco Idiart, and Aline Villavicencio. 2019. The BRWAC corpus: A new open resource for Brazilian Portuguese. In *LREC 2018 - 11th International Conference on Language Resources and Evaluation*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. https://doi.org/10.18653/v1/n18-1101 arXiv:1704.05426