

TSAH DU: Open Source Expert System for Law, Vector Representation of Meaning in Judicial Searches

Waa jacu^{a,*}

^aSantiago Restrepo Ruiz, Colombia.

ARTICLE INFO

Keywords:

Law, Open Source, Search Engine, Expert System, Transformers, Natural Language Processing, Neural Networks, NLP, NLU, Colombia.

ABSTRACT

Non weaponizable research the sort of Law and Market requires caution and broadcasting easy to use interfaces. Market is being still stabilized for a broad and stable theoretical common fortune of all commendable participants. Effective and broad access to Law however is intended in this paper; To be distributed and used without cost: <https://github.com/savethebeesandseeds/tsahdu>

1. Introduction

Exercise a natural advice of caution those doing advanced Mathematics, an assertive guide not to risk a weaponizable tool. Laws sort a encyclopedic strength, prerogative of these with aims to reduce the barrier of knowledge and the price of legal advice; a tool on perpetuity free and with open access. The method serves to consult the state of Law and the state of Jurisprudence, given a Text describing a situation; Tsahdu is a modern search engine dedicated to Law. Implementation is for now limited to the sovereign Constitution and the sovereign Codes of Law of the Republic of Colombia.

2. Mathematical Background

Used basis are, real numbers: \mathbb{R} , integers: \mathbb{Z} . To represent a phrases in Human language: \mathbb{W} . And a reference in a document of Law: \mathbb{Q} . Here a scalar v is specified to be real if $\mathbb{R}v$ and a p -dimensional real vector is defined as: $\mathbb{R}^p V$. Multiple vectors are grouped like Tensors, m grouped vectors is defined as $\mathbb{R}^p V^m$. Reference to a *document of Law* discriminates n individual Articles $\mathbb{W}\mathcal{A}^n$.

Following a static transformation $\phi(\cdot)$ each article is represented by a dense vector:


$$\phi(\mathbb{W}\mathcal{A}^n) = \mathbb{R}^d V^n. \quad (1)$$

Where \mathbf{V} is known as Embedding (in common transformation models $d = 384$), $\mathbb{W}\mathcal{A}$ are Article Paragraphs of varying length up to $s = 512$ words long, n is a positive integer $^+\mathbb{Z}$ specifying the article index.

Considering any paragraph or phrase is transformed into a dense vector $\{\mathcal{A}^n \rightarrow \mathbf{V}^n\}$, a document of Law is understood as a dense Matrix $\mathbb{R}^n \times \mathbb{R}^d \mathbf{M}$ of size $n \times d$. Any article larger than $s = 512$ words is fragmented and averaged in vector space.

The aforementioned transformation to vector space $\phi(\cdot)$ is done using [1] implementations: Natural Language Transformers Models¹, on what is known as a Sentence Transformer². Whereby two sentences are alike if their vector representation is alike.

*Corresponding author

 savethebeesandseeds@gmail.com (Waa jacu)

 www.waa jacu.com (Waa jacu)

ORCID(s): 0000-0001-6386-9350 (Waa jacu)

¹<https://huggingface.co/>

²<https://huggingface.co/sentence-transformers/>

3. An Expert System for the Law

Search engines can be modular or anyway very complex, mixing multiple searching methods. This expert system description considers only the comparison in vector space.

A citizen needing advice of Law will query a description paragraph $\mathbb{W}\mathcal{U}$, the expert system will consider the vector coming from the query transformation $\phi(\mathbb{W}\mathcal{U})$ and compare it against the matrix of Law \mathbf{M} . Rows most similar with the user query are returned as a reference of Law, note that every row in \mathbf{M} has associated a specific reference of Law \mathbb{Q} , e.g. article number n in chapter q , title u , book w and document z , this is $\mathbb{Q} :< z, w, u, q, n >$.

Regex expressions are used to separate Articles, discriminate references. There is generally one Constitution and many Laws compiled in documents known as Codes. Robots automates the downloading and updating of those .pdf documents. Further automation extracts the text content of the document and generates regular .txt files; Finally using regular Expressions (regex) the full text is separated into individual articles with their associated reference.

4. An Expert System for Jurisprudence

These are Court sentences and resolutions, serves an enormous advantage to the exercise of a Judge and required to provide a proper trial. Also to evaluate the cogency for the exercise of Law in a sovereign state.

Defining a reference to a sentence of a Court dictum on *Jurisprudence* searches requires the same vector approach $\phi(\cdot)$ but a different indexing of phrases, as the division by articles is only found in documents of Law. To generalize the technique across different type of documents, the role of n is changed to the role of ρ . Where ρ is a positive integer $^+\mathbb{Z}$ know as the document density count, it varies from document to document as different documents has different sizes — for example, $\rho = 100$ implies a document was fragmented into 100 phrases. Creating, for every document a matrix $\mathbb{R}^\rho \times \mathbb{R}^d \mathbf{M}$.

5. Considerations of Implementation

Law recommendation is a sensible topic, an online open service requires an explicit caution note specifying a check statement: "[this is not a professional advice of law](#)".

The colossal body of Jurisprudence and the monumental size of Law, require computational efficiency of implementation, so a high performance programming language is recommended. Robotic Web Process Automation is probably required to scrap and update Web documents. Transformers Networks are downloaded pretrained for the idiom or matching tongue, further fine tuning is recommended.

The sensibility of the topic involve require the Transformer Network be trained properly, on a mistake in a case of Law the developers intend to show good faith by maximizing likelihood on the training procedure.

.pdf documents are better stored as .txt files, not every .pdf document has a straightforward way to transform into a .txt, sometimes software known as optical character recognition is required.

Regex as a way to segment files into articles is not very stable, on this part there is still a lot of Human work required.

6. Vector Representation of a Phrase

Law is very well written, but natural language is not precise; The challenge for English is the same challenge for Spanish, they have not a straightforward analytical representation $f(\cdot) : \mathbb{W} \rightarrow \mathbb{R}^x$. Best I know how is Deep Learning Transformers $\phi(\cdot) : \mathbb{W} \rightarrow \mathbb{R}^d$, as a way to formulate Text (\mathbb{W}) as a precise numeric abstraction (\mathbb{R}).

A high goal for the sciences of language is to construct and agree on an analytical transformation of idioms to vectors, an orthogonal language would serve Humanity with great progress, but it is proven to be very hard, justice is mixed with poetry and without extensive context science seems like fantasy.

Until a transformation is discovered and accord on, best we can do is to use Deep Neural Networks, generally known as Text Input Transformers Neural Architecture; Where a specific Layer inside the Deep model is used to represent the Input Phrase as a Vector of embedding.

For English Language: *all-MiniLM-L12-v2*³ with a vector size $d = 384$.

For Spanish Language: *sentence_similarity_spanish_es*⁴ with a vector size $d = 768$.

There are Transformers for other languages; English Model performs better than Spanish. With a sufficient corpus of Law one can fine train any of these Models to better perform.

7. Stable Embedding

These are some marginal operations to better represent the embedding vector: $\mathbb{R}^d \mathbf{V}$. It be useful for the vector to be normalized, two types of normalization are consider:

$$\mu_1 - \text{normalization} := \mathbb{R}^d \mathbf{V} = \frac{1}{\mathbb{R}\mu_1} \cdot \mathbb{R}^d \mathbf{V}, \quad \text{where : } \mathbb{R}\mu_1 = \sum_{i=1}^d \mathbb{R} \mathbf{V}_i \quad (2)$$

$$\mu_2 - \text{normalization} := \mathbb{R}^d \mathbf{V} = \frac{1}{\mathbb{R}\mu_2} \cdot \mathbb{R}^d \mathbf{V}, \quad \text{where : } \mathbb{R}\mu_2 = \sum_{i=1}^d \text{abs}(\mathbb{R} \mathbf{V}_i) \quad (3)$$

$$\mu_3 - \text{normalization} := \mathbb{R}^d \mathbf{V} = \frac{1}{\mathbb{R}\mu_3} \cdot \mathbb{R}^d \mathbf{V}, \quad \text{where : } \mathbb{R}\mu_3 = \sqrt{\sum_{i=1}^d (\mathbb{R} \mathbf{V}_i)^2} \quad (4)$$

Where (\cdot) is the element-wise multiplication.

From now on, unless otherwise mention assume \mathbf{V} and all the rows in \mathbf{M} are $\mu_3 - \text{normalized}$, this is the standard way to normalize a vector.

8. Average vector

A policy for dealing with long phrases is superposition of rolling overlapping windows of phrase segments averaged in vector space. The superposition of multiple (m) vectors outputs a vector:

$$\mathbb{R}^d \text{Avg}(\mathbf{V}^m) = \frac{1}{m} \cdot \sum_{i=1}^m \mathbb{R}^d \mathbf{V}^{(i)} \quad (5)$$

Where (\cdot) is the element-wise multiplication, and Σ sums vectors.

In Law, Articles are grouped in Chapters, Chapters are grouped in Titles and all Titles make a Document. It is useful to know the average vector of an entire Chapter ($\mathbb{W}\mathcal{C}$), Title ($\mathbb{W}\mathcal{T}$), Book ($\mathbb{W}\mathcal{B}$) or Document ($\mathbb{W}\mathcal{D}$), and for long Articles also ($\mathbb{W}\mathcal{A}$). —the vector of it's member phrases average, and so on,

$$\begin{aligned} \mathbb{R}^d \phi(\mathbb{W}\mathcal{A}^n) &= \text{Avg}(\mathbf{V}^m), \quad \forall m \in \mathcal{A}^n, & \text{average vector over all phrases in the } n\text{th-article} \\ \mathbb{R}^d \phi(\mathbb{W}\mathcal{C}^q) &= \text{Avg}(\mathbf{V}^m), \quad \forall m \in \mathcal{C}^q, & \text{average vector over all phrases in the } q\text{th-chapter} \\ \mathbb{R}^d \phi(\mathbb{W}\mathcal{T}^u) &= \text{Avg}(\mathbf{V}^m), \quad \forall m \in \mathcal{T}^u, & \text{average vector over all phrases in the } u\text{th-title} \\ \mathbb{R}^d \phi(\mathbb{W}\mathcal{B}^w) &= \text{Avg}(\mathbf{V}^m), \quad \forall m \in \mathcal{B}^w, & \text{average vector over all phrases in the } w\text{th-book} \\ \mathbb{R}^d \phi(\mathbb{W}\mathcal{D}) &= \text{Avg}(\mathbf{V}^m), \quad \forall m. & \text{average vector over all phrases in the document} \end{aligned} \quad (6)$$

³<https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2>

⁴https://huggingface.co/hiiamsid/sentence_similarity_spanish_es

Those are meant to be simple formulas, but imply some considerations; let me explain by a simple example: Imagine a window size of $s = 2$ words and overlapping of size $g = 1$, the user inputs this phrase:
 \mathbb{w} "Dogs chase Cats" $\rightarrow [\mathbb{w}$ "Dogs chase", \mathbb{w} "chase Cats"] = $[\mathbb{R}^d[2.0, 3.0, 1.0, \dots], \mathbb{R}^d[4.0, 5.0, 6.0, \dots]]$,
 \mathbb{w} "Dogs chase Cats" = $\mathbb{R}^d[3.0, 4.0, 3.5, \dots]$

Since the phrase was too long, the overlapping window policy segmented the input phrase into two partial phrases. The result is the average vector, consider to think the embedding for the word in the middle \mathbb{w} "chase" was considered two times, this is not ideal.

If $s = 2$ and there is no overlap considered $g = 0$, contextual information might be broken:
 \mathbb{w} "Dogs chase Cats" $\rightarrow [\mathbb{w}$ "Dogs chase", \mathbb{w} "Cats"] = $[\mathbb{R}^d[2.0, 3.0, 1.0, \dots], \mathbb{R}^d[1.0, 0.5, 9.0, \dots]]$,
 \mathbb{w} "Dogs chase Cats" = $\mathbb{R}^d[1.5, 1.75, 5.0, \dots]$

The two approaches yield different average vectors, this trade-off is hard to come-by.
 Values of $s = 512$ and $g = 64$, seem to work fine.

9. Dense Vectors Comparison

Be sure any vector \mathbf{V} is normalized in it self: $\mathbf{V} = \|\mathbf{V}^*\|^{-1} \cdot \mathbf{V}^*$ so that, $\sum_{j=1}^d V_j = 1$.
 Comparing two vectors, vector $(\mathbb{R}^d \mathbf{V}_a = a)$ and $(\mathbb{R}^d \mathbf{V}_b = b)$, these are some simple comparison formulas:

$$\begin{aligned}
 \text{Minkowski :} & \quad \left(\sum_{j=1}^d (a_j - b_j)^p \right)^{1/p} \\
 \text{Cosine :} & \quad \frac{\sum_{j=1}^d a_j \cdot b_j}{\sqrt{\sum_{j=1}^d (a_j)^2} \cdot \sqrt{\sum_{j=1}^d (b_j)^2}} \\
 \text{Dot - Product :} & \quad \sum_{j=1}^d a_j \cdot b_j
 \end{aligned} \tag{7}$$

Where $\mathbb{R} a_j$ and $\mathbb{R} b_j$ are the scalar components in the j index of the vectors $\mathbb{R}^d \mathbf{V}_a$ and $\mathbb{R}^d \mathbf{V}_b$ respectably. For Dot-Product and Cosine, the higher the value the better, for Minkowski the lower the value the better. Euclidean distance, arise from Minkowski where $p = 2$. However, the size for the dense vector is very high $d = 384$, the overall information from the upmost relevant dimensions might be opaqued given the vector size, to illustrate this let's view in the Figure 1 below:

About dense high dimensional vector comparison one might go beyond simple analytical formulas; to train a machine learning comparison schema, derive Principal Component Analysis or Relevance Vector Machines to reduce the d -dimensional of the vectors. Or describe optimization procedures, label or in anyway train a Model to compare vectors. For now the Euclidean distance suffice, on these sensible topics efforts rather help to construct analytical solutions, as it is hard to argue the validity of a solution involving deep compositional function approximations, like Neural Networks.

To end the section, let me be simple about it; if every Article is transform into a Vector and a corpus of Law is a Matrix. A user input is transformed and compared against the matrix of Law \mathbf{M} , returning the references of Law corresponding to the rows indices with the lower distance.

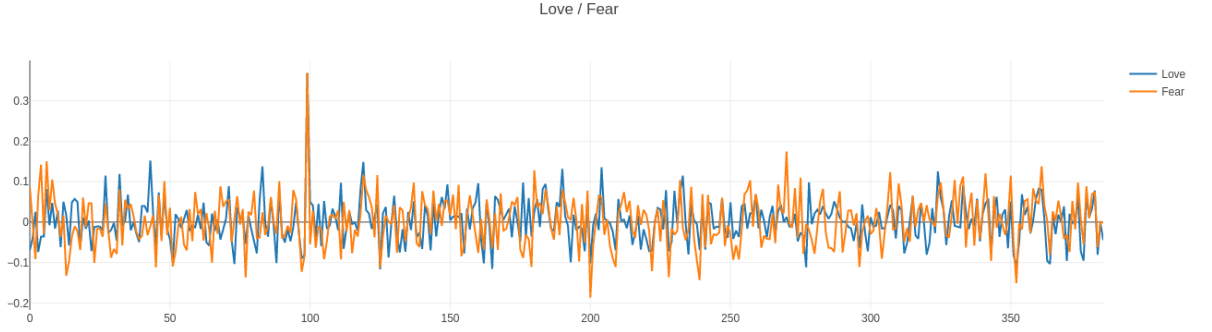


Figure 1: Embedding Vectors for the words **Love** and **Fear**, To be notice index 99 might be a flag for the Human emotion component. The numeric comparison yields are: Minkowski(p=1): 0.13416162, Minkowski(p=2): 1.0309231, Minkowski(p=3): 0.21693337, Minkowski(p=5): 0.19361636, Cosine: 0.4685991 [rad]. Dot-Product: 0.4685991. These two ideas are as expected fairly different, as the vectors are μ_3 – *normalized* the maximum possible distance using Minkowski(p=2) is 2.0.

10. Entropy, Information of the Vector Space

In this framework, Entropy of a Text Fragment is the Entropy of a Layer in a Natural Language Transformer Neural Network. This is to say, Entropy of a Text is for this framework the Entropy of the Embedding Vector.

Entropy might be in units of [Shannon] (or [bits]) if \log_2 is used, or in units of [nats] if \log_e . Let's use the [Shannon] terminology to share respect. Vector Space Entropy usefulness insights about the information of a given fragment of Text depends on the method used to train the Transformer Network. Experiments for the used Transformer yield **not determinant results**.

A way to calculate Entropy of a Phrase in Vector Space is shown, requires three steps:

Assert that the Vector is μ_3 – *normalized*.

Have the Vector resemble a Discrete Distribution, on this case let's simply calculate the absolute value in all the dimensions of the vector:

$$\mathbb{R}^d \mathbf{V}_j^* = \text{abs}(\mathbb{R}^d \mathbf{V}_j) \quad (8)$$

Finally the Entropy is:

$$\mathbb{R} \mathbf{H}_\mathbf{V} = (-1) \cdot \sum_{j=1}^d \mathbf{V}_j^* * \log_2(\mathbf{V}_j^*) \quad [\text{Shannon}] \quad (9)$$

Again, this yield **not determinant results**. The problem is that the Dense vector is defined over positive and negative real numbers, a better way to perform the operation in equation 8 will unlock the whole science of information theory. On research I try on with building parametric distributions, and also try on having the negative values be a separated distribution and compute the join Entropy, but had no significant results. Maybe the solution be on changing the Network Architecture and training with Cross Entropy.

11. Results

Implemented in Rustc Language for the Colombian Constitution and for the Colombian Codes of Law in the Republic. Source code is available with open access at <https://github.com/savethebeesandseeds/tsahdu>. A demo server was mounted and hosted locally <http://www.waajacu.com>.

Achieve a high velocity and a stable implementation using a Spanish Language Transformer `sentence_similarity_spanish_es` with a maximum phrase size of $s = 512$ words and overlapping of $g = 64$ words.

Complications on gathering the data, .txt files for all Codes of Law and .txt file Article segmentation.

12. Future Work

Refine the current work, implement for other languages and for other countries. A better comprehension for the Entropy of the Vector Space will suffice the base for all the information analytics.

13. Acknowledgments

This work was not granted any private or public founds. Research and development was intended for non profit and was done by Waajacu. All the mistakes are my own, to be acknowledge are the Mathematicians who developed the state of the Art for Natural Language Understanding and the cooperation of Machines.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.