

# Representing Contracts by Clause Genome for Machine Learning Applications

**Abstract.** Contracts drive large scale commercial transactions. Being a legal document it is binding to all the parties, stakeholders involved. Composition contracts using clauses in an unambiguous manner is key to a good contract design. Contract analysis becomes critical in case of already drafted contract for detecting important clauses, obligations, key attributes, etc. Artificial Intelligence, in form of Machine Learning is being used effectively in some of these critical requirements. Finding similar contracts, grouping them is one such specific example.

Text needs to be converted to numbers in order to be used for Machine Learning applications. Various such vectorization techniques are available, which are primarily statistical in nature and deal with words in the documents. Word vectors are then composed into respective document vectors. This paper proposes vectorization of a higher level abstraction, called clauses, there by bringing domain specific semantic composition into vectorization. Efficacy of the proposed approach has been demonstrated using Similarity and Clustering algorithms.

**Keywords:** Clause Genome, Contract Categorization, Representation Learning

**DOI:** <https://doi.org/10.14733/cadaps.2022.aaa-bbb>

## 1 INTRODUCTION

Machine Learning approaches need inputs to be in numeric form. For Natural Language Processing (NLP), it becomes imperative to convert input (and output, in some cases) to numeric form, such as vectors. Multiple approaches to map words or sentences or document to vectors are available. Some popular ones such as 'Bag of Words (BoW)', 'Term-Frequency Inverse Document Frequency (Tf-idf)' are typically word frequency based, whereas Word Embeddings such as 'Document to Vectors (Doc2Vec)', Glove, 'Bidirectional Encoder Representations from Transformers (BERT)' are co-occurrence or contextual in nature. Domain specific embeddings tend to capture some form of semantics due to co-occurrences, ie vectors associated words are close in embeddings space [1].

But such word level vectors are too granular to capture higher level domain constructs or abstractions. For example, in case of legal contracts, Doc2Vec would be some sort of composition (say, averaging) of Word2Vec of individual words. This approach loses middle level abstractions such as clauses, ie. Contract is composed of clauses, clauses are composed of words. The middle level abstract actually captures legal-domain semantics.

This paper proposes a novel approach of represent contracts by vector of clauses. Once documents are converted to list of clause category labels, its a string of labels, similar to biological genome made up of letters such as A, T, C, G. Thus the clause category based representation of contract document is referred to as 'Clause Genome'. They act as signature or short form string of the original document, which can further be vectorized using traditional approaches mentioned above, and then used to Machine Learning algorithms such as Classification, Clustering, Similarity, etc.

## 2 RELATED WORK

TBD

### 3 PROPOSED APPROACH

The proposed approach takes document corpus as input, converts each document into “Clause Genome” based intermediate representation, which then taken further into standard Text Classification pipelines such as Vectorization, Classification model training and results generation. Major steps are enumerated below:

- Data Preparation: Documents, if non-textual format, are converted to text. Text is segmented into paragraphs.
- Clause Classification: Each paragraph gets classified into a clause category, thus a document becomes list of clause categories.
- Symbolization: For each document, using list of clause categories, a “Clause Genome” sequence or string is generated. This shortened form now represents the original document in a shortened form.
- Vectorization: Standard vectorizers like BoW or TfIDF are used on the shortened “Clause Genome” string.
- Application: Genome based vectors can be used in Machine Learning tasks such as Classification and Clustering.

Core steps have been explained in detail, in the following sub sections.

#### 3.1 Symbolization

Typical Master Services agreement will have about 70-100 categories of clauses. Machine Learning based approaches such as Support Vector Machines, Naive Bayes, Neural Networks, can be used to build a classifier.

After ‘Data Preparation’ and ‘Clause Classification’ phases, each document is in the form of list of clause categories in it, while maintaining the order of the clause sequence. Each category is mapped to a symbol or label. It can be letters like ‘a’, ‘b’, ‘c’, etc. But to accommodated large number of categorizes, labeling scheme as ‘c1’, ‘c2’, ... is used. Sample symbolization map is shown below:

```
clause_category_to_symbol_map = {
    "acceptance": "c1",
    "publicity": "c2",
    "amendment": "c3",
    :
    "intellectual_property_rights": "c9",
    "penalties/remedies": "c10",
    "business_continuity": "c11",
    :
    "payment_terms": "c24",
    "force_majeure": "c25",
    "governing_law": "c26",
    :
    "quality_control": "c60",
    "preamble": "c61",
    :
}
```

For Sample documents like MSA\_1 and MSA\_2, the clause genomes look like:

```
MSA_1 = 'c61 c53 c3 c40 c23 c7 c8 c40 c8 c1 c40 c40 c1 c20 c24 c24 c47 c24 c24 c50 c7 c7 c7 '
MSA_2 = 'c61 c61 c48 c48 c8 c34 c40 c40 c8 c24 c47 c1 c49 c14 c29 c2 c30 c24 c30 c9 c53 '
```

Note that both seem to have started with similar clause category, "c61", and thats obviously "preamble". Pictorary they would look as in Fig. 1:



**Figure 1:** Graphical representation of document clause genomes

### 3.2 Vectorization

Instead of the original text, now treat documents are just the genome strings. Use this new content to vectorize them, tfidf in the following case as shown in Fig. refgenomevectors.

document	filename	x_0	x_1	x_10	x_11	x_12	x_13	x_14	x_15	x_16	x_17	x_18	x_19	x_2	x_20	x_21	x_22
c61	c61 c4 MSA 1.txt	0.125136	0	0.062568	0	0.062568	0	0.180799	0	0.055831	0	0.040847	0.062568	0	0	0.637327	0
c61	c53 c3 MSA 2.txt	0.057626	0.010002	0.007684	0.027425	0.015367	0.004348	0.233127	0.015002	0.013712	0.013044	0.010032	0.011525	0.005921	0.010002	0.069569	0.011842
c61	c53 c2 MSA 3.txt	0.219677	0	0.043935	0.078409	0.043935	0	0.285654	0.343145	0.039204	0.099452	0.057366	0.043935	0	0.171573	0	0
c14	c8 c14 NDA 1.txt	0	0	0	0.041296	0	0	0	0	0.041296	0	0.09064	0	0	0	0	0
c61	c53 c2 NDA 2.txt	0	0	0	0	0.183431	0	0.132512	0	0.163679	0	0.119752	0	0	0	0	0
c61	c43 c1 NDA 3.txt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c61	c53 c5 SOW 1.txt	0	0.030326	0.093189	0.062366	0	0.052735	0.185132	0	0	0.026368	0.030419	0	0	0	0	0
c17	c17 c1 SOW 2.txt	0.11505	0	0	0.025665	0	0.032553	0.644125	0	0	0	0.037555	0.028762	0	0	0.032553	0
c48	c40 c4 SOW 3.txt	0	0	0	0	0	0	0.526934	0	0	0	0.238096	0	0	0	0	0

**Figure 2:** Vector representation of document clause genomes

### 3.3 Applications

Efficacy of the 'Clause Genome' based approach for Similarity calculations can be demonstrated by checking two Master Service Agreements (MSA) and then checking between one MSA and one NDA (Non Disclosure Agreement). Results are:

- Cosine Similarity : MSA 1 and MSA 2: 0.734940271290997
- Cosine Similarity : MSA 1 and NDA 1: 0.31642358837691426

It can be clearly seen that the similar type of documents have high and dissimilar ones have low similarity score. Meaning, clause genome representation is quantitatively in agreement with the document language.

Results for checking the efficacy of the 'Clause Genome' based approach for Clustering is:

Barring one misclassification, documents have got correctly segregated into their own type clusters.

## 4 CONCLUSIONS

TBD

- Advantages:
  - Leverages Contract domain specific higher level abstractions, making it more domain-semantic.
  - Clause genome has lowr space requirements and computationally efficient for firther downstream applications.
  - Noise based on individual word frequency is smoothened
  - Order of the clauses is maintained ie document structure is retained

AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH
x_54	x_55	x_6	x_7	x_8	x_9	cluster id	distance f	cluster id	filename
0.365028	0.100981	0	0.784306	0	0	0	0.580109	0	SOW 1.txt
0	0	0	0.215402	0	0	0	0.580109	0	SOW 3.txt
0.742389	0.033304	0.003842	0.099836	0.005921	0.005001	1	0.417208	1	MSA 2.txt
0.458928	0.126957	0.087871	0.25949	0	0	1	0.467726	1	MSA 3.txt
0.300438	0.041556	0.028762	0.186863	0	0	1	0.536209	1	SOW 2.txt
0.531013	0.135599	0	0.073907	0	0	1	0.652684	1	MSA 1.txt
0.359256	0.132512	0.183431	0.108337	0	0	2	0.332809	2	NDA 2.txt
0.18128	0	0	0.054667	0	0.060242	2	0.429916	2	NDA 1.txt
0.748489	0.138041	0	0.338572	0	0	2	0.556984	2	NDA 3.txt

**Figure 3:** Vector representation of document clause genomes

- Explainability is better for debugging as well as customer trust
- Disadvantages:
  - Heavily depends on Clause Classifier efficacy
  - Disregards variations within a clause category
  - Loss of word-level information including linguistic structures.

## REFERENCES

- [1] Palachy, S.: Document embedding techniques, 2019. <https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d>.