# Extracting Relevant Sentences from Past Court Decisions: An Important First Step of A Legal Deep Learning Research Project

Wai Yin Mok and Jonathan R. Mok

***Abstract*—This paper presents a potential solution to the problem of extracting relevant sentences from past court decisions, which is an important first step of our legal deep learning research project. Court decisions are typically written in natural language like English. Hence, our extraction solution first uses legal statutes to construct an ontology for the desired sentences, and then uses NLTK (Natural Language Toolkit), a Python Natural Language Processing Toolkit, to construct search patterns based on the ontology to extract relevant passages from hundreds or thousands of past court decisions. The extracted sentences will be further processed and the resulting information will then be fed into a deep learning system, whose purpose is to assist legal practitioners by selecting relevant documents and streamline litigation.**

***Index Terms*—Legal deep learning research project, ontology, nltk (natural language processing toolkit), tokens, stemmer, semantic similarity, wordnet.**

## I. Introduction

The ultimate goal of our legal deep learning research project currently undertaken at The University of Alabama in Huntsville is to develop an automated support system that assists legal practitioners in selecting relevant documents and streamlines litigation. In addition to developing traditional decision trees, which have been used successfully in many other data mining applications, this research project also constructs a deep learning system that analyses hundreds, if not thousands, of past court decisions that are similar to the law suit at hand. Looking for similar precedents is a common first step taken by most legal practitioners when faced with a new case; doing so allows practitioners to better predict the outcome of a new case and allows them to hone in on relevant documents from among hundreds or thousands of others to build their case. Automating such an analysis of past court decisions by computers can certainly help increase search space and reduce search time.

Although such a proposed system holds great promise, many obstacles need to be first overcome for it to become a reality. This paper presents a potential solution for an important first step of this system: extracting relevant passages from past court decisions. Past court decisions are typically written in natural language like English. Hence, a software component that is able to process written texts and extract the relevant information thereof is an essential element. In recent years, natural language processing has come a long way. It is common nowadays to see automated systems responding fairly well to humans. In addition, to some degree of success, major search engines are also able to understand search items entered in free texts by humans. Based on years of linguistic and AI researches, mature natural language processing toolkits are also available. A notable example is NLTK (Natural Language Toolkit), which can be downloaded for free at http://www.nltk.org/ [1], [8].

Besides NLTK, a second essential step for our extraction solution is to develop an ontology of the desired information. An ontology is a conceptual model of the targeted information, which specifies the structure and the attributes thereof. The ontology can then be used to form a search template. Ontological search has been successfully applied in many areas, including family history and biology [2]-[7]. Based on the ontology, search patterns are created, which will be applied in the search written in Python code using the NLTK module.

This paper is organized as follows. We first demonstrate the ontology used in this paper. We then show the search patterns derived from the ontology. After which, conclusions will be presented.

## II. Creating an Ontology

This paper uses a breach of contract court decision as an example. Breach of contract can happen in many ways, and nonconforming goods is a type of breach of contract. Furthermore, court decisions on nonconforming goods look to statutory laws that govern how such goods are treated. The example used in this paper is based on the statute Code of Alabama Section 7-2-601, which provides buyer's rights regarding nonconforming goods. Fig. 1 shows an ontology created from that statute. A rectangle in the figure represents the elements belonging to the set whose name is written inside of the rectangle. For example, the rectangle "breach of contract" represents a set of court decisions involving breach of contract. An open triangle denotes a is-a relationship. Hence, a court decision on buyer's rights regarding nonconforming goods is also a breach of contract court decision and also looks towards Alabama statutory law. The rectangle with the name Code of Ala. §7-2-601 represents the set of court decisions that look towards this section. Furthermore a §7-2-601 court decision has many components: one or more plaintiffs, one or more defendants, and one or more contracts, which are also shown in Fig. 1. Each contract

Manuscript received February 10, 2018; revised May 5, 2018.

Wai Yin Mok is with Department of Information Systems, The University of Alabama in Huntsville, Huntsville, Alabama, 35899, USA (e-mail: mokw@uah.edu).

Jonathan R. Mok was with School of Law, The University of Alabama Houston, Tuscaloosa, AL 35487, USA (e-mail: Jon.mok@law.ua.edu).

doi: 10.18178/ijke.2018.4.1.094

is further related to one or more disputes and each dispute is associated with one or more facts and one or more reasonings, which are explanations of how the court applies the law to the facts.

Although each set of elements in Fig. 1 represents a legal concept, each set is nevertheless associated with one or more legal keywords in a court decision. For example, the set of contracts is associated with the keyword "contract." The keywords associated with each set of elements will be used to extract the desired sentences from the court decision.

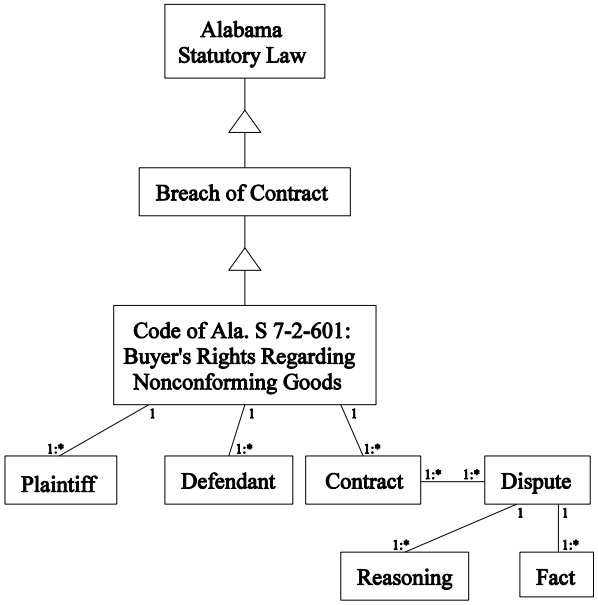



Fig. 1. An ontology created from code of Ala. S 7-2-601, which specifies the structure and the attributes of the targeted information.

## III. Ontological Search

### A. NLTK (Natural Language Toolkit)

This section presents the basic steps that we took in using NLTK, a popular natural language processing toolkit, for this paper

```
import nltk

raw = open('case.txt','r').read()
caseTokens = nltk.word_tokenize(raw)
caseTagged = nltk.pos_tag(caseTokens)
stemmer = nltk.PorterStemmer()
caseStemmed = list(map(stemmer.stem,caseTokens))
```

Fig. 2. The basic NLTK setup steps of this paper.

The case file was first opened and then a list of tokens called caseTokens was generated. Each token was further tagged with its type, which was stored in the list called caseTagged. Some examples are ('Opinion', 'NN'), ('by', 'IN'), (':', ':'), ('SHORES', 'NNP'), where 'Opinion' is a NN (noun), 'by' is an IN (preposition), ':' is a colon, and 'SHORES' is a NNP (proper noun). The stemmer reduces each token to its stem (root) before the keywords are searched in the case file. For example, 'Contract', 'contract', 'contracts' are all reduced to 'contract'. Reducing each token to its stem increases the number of matches in the case file.

### B. An Example Court Decision

The particular breach of contract court decision used in this paper is Gulf Coast Fabricators, Inc. v. Mosley, 439 So. 2d 36 (Ala. 1983), written by the Supreme Court of Alabama on September 23, 1983. We obtained this court decision through LexisNexis®, which added its own annotations. Hence, our first step was to remove the annotations of LexisNexis®. The actual court opinion starts with the keyword "opinion." Hence, our next step was to locate the token 'opinion' in the list caseTokens.

```
def find(aSList,aBList,start=0):
    pos = []
    stemmed = list(map(stemmer.stem,aSList))
    for i in range(start,len(aBList)-
                   len(aSList)):
        if stemmed ==
               caseStemmed[i:i+len(aSList)]:
            pos.append(i)
    return pos

opinionPos =
find(nltk.word_tokenize('opinion'),caseTokens)
```

Fig. 3. The function find, whose purpose is to locate a token within the list caseTokens.

The function find is used to locate a particular token in a given list. The result of the function call is that the token 'opinion' is at the index 792 in the list caseTokens. This means that all searches will be done on or after index 792.

### C. Semantic Similarity

Although each set of elements in the ontology of Fig. 1 is associated with one or more keywords, it is possible that there might be other words in the example court decision that have similar meanings but are not explicitly specified. Hence, we must first discover them in the case file. For this paper we used the following code snippet to locate words that are similar to "contract". The method similar provided by nltk.Text finds other words that appear in the same contexts as the specified word. The resulting words found by the method similar are shown in Fig. 5.

```
caseText = nltk.Text(caseTokens)
print(caseText)
caseText.similar('contract')
```

Fig. 4. Finding similar words to "contract" by the method similar, which is based on distributional similarity.

```
building record fact gives slab case on tender buyer
findings gcf agreement size labor storing sale whether
```

Fig. 5. The resulting similar words to "contract" in the example case file.

Some words in Fig. 5 are clearly not similar to "contract." Nevertheless, NLTK has a corpus reader called "WordNet" (http://www.nltk.org/howto/wordnet.html), whose interface contains a method called "wup_similarity" that can be used to compute the similarity of two words. As an example, the words "contract" and "agreement" were compared in the following snippet and the results are shown in Fig. 7.

```
from nltk.corpus import wordnet as wn
contractSynsets = [ss for ss in
    wn.synsets('contract','n')]
print(contractSynsets)
agreementSynsets = [ss for ss in
wn.synsets('agreement','n')]
```

```
  print(agreementSynsets)

  pairs = [(c.wup_similarity(a),c,a) for c in
   contractSynsets for a in agreementSynsets]
  match = max(pairs)
  print(match[0],match[1],match[2])
  print(match[0],match[1].lemma_names(),match[2].l
emma_names())
```

Fig. 6. Checking similarity of the words “contract” and “agreement.”

```
   [Synset('contract.n.01'),
Synset('contract.n.02'), Synset('contract.n.03')]
  [Synset('agreement.n.01'),
Synset('agreement.n.02'), Synset('agreement.n.03'),
Synset('agreement.n.04'), Synset('agreement.n.05'),
Synset('agreement.n.06')]
  0.8571428571428571       Synset('contract.n.01')
Synset('agreement.n.01')
  0.8571428571428571   ['contract']   ['agreement',
'understanding']
```

Fig. 7. The results of checking the meanings of the words “contract” and “agreement.”.

Considering them as nouns, WordNet specifies that “contract” and “agreement” respectively have three and six different meanings. However, Synset('contract.n.01') and Synset('agreement.n.01'), the most common meanings of “contract” and “agreement,” have the greatest score 0.8571 of similarity. Because of such a high score, the word “agreement” was added to the keywords associated with the set “contract” in Fig. 1. In addition, since the word “agree” is the verb form of “agreement,” the word “agree’ was also added as well. The other sets of elements of the ontology in Fig. 1 also went through this same process of identifying additional keywords.

### *D. Finding Relevant Sentences*

After determining the keywords for each set of elements in the ontology in Fig. 1, the function find was used to locate the appearances of each keyword in the case file. Note that the function find first reduces each word to its stem before equality comparison is made, thus maximizing the number of matches. Using the keywords “contract, “agreement” and “agree” as an example, the following code snippets were generated.

```
  keyWords = ['contract','agreement','agree']
  keyWordTokens                                  =
list(map(nltk.word_tokenize,keyWords))
  keyWordPos = [(w,find(w,caseTokens,opinionPos[0]))
for w in keyWordTokens if
              find(w,caseTokens,opinionPos[0])   !=
[]]
  keyWordPos = sorted(keyWordPos,key=lambda x: x[1])

  allPositions = sorted([pos for w in keyWordPos for
pos in w[1]])
```

Fig. 8. Searching the keywords “contract,” “agreement,” and “agree” in the example case file.

The list keyWordPos contains the indexes of the keywords, which are shown as follows:

```
  [(['contract'], [836, 1107, 1115, 1153, 1315, 1331,
1359, 1583, 1591, 1718, 1842, 1854, 1866, 1900, 2259,
2274, 2307, 2326, 2340]), (['agreement'], [856, 1585,
1598, 2205]), (['agree'], [875, 1047, 1651, 1672])]
```

Fig. 9. The indexes, or positions, of the appearances of the keywords.

Another function named sentence is used to find the sentences that contain the appearances of the keywords.

```
def sentence(pos,retList=caseTokens):
 length = len(caseTagged)
 stop = length-1
 start = 0
 for k in range(pos+1,length):
     if caseTagged[k][1] == '.':
         stop = k
         break
 for k in range(pos-1,-1,-1):
     if caseTagged[k][1] == '.':
         start = k+1
         break
    return(retList[start:stop+1])
```

Fig. 10. The function sentence, whose purpose is to retrieve the sentence that includes the token at index pos.

Some of the sentences found by the function sentence are shown below. In addition to the sentence, we also include the index of the keyword in the sentence and the two preceding and two succeeding tokens.

```
  875 -> Mosley also agreed to perform
  [('Mosley', 'NNP'), ('also', 'RB'), ('agreed',
'VBD'), ('to', 'TO'), ('perform', 'VB'), ('all',
'DT'), ('necessary', 'JJ'), ('slab', 'NN'), ('and',
'CC'), ('foundation', 'NN'), ('work', 'NN'), ('for',
'IN'), ('the', 'DT'), ('new', 'JJ'), ('building',
'NN'), ('.', '.')]

  1047-> building and agreed to make
  [('GCF', 'NNP'), ('instructed', 'VBD'), ('Mosley',
'NNP'), ('to', 'TO'), ('proceed', 'VB'), ('with',
'IN'), ('construction', 'NN'), ('of', 'IN'), ('the',
'DT'), ('new', 'JJ'), ('building', 'NN'), ('and',
'CC'), ('agreed', 'VBD'), ('to', 'TO'), ('make',
'VB'), ('progress', 'NN'), ('payments', 'NNS'), ('to',
'TO'), ('Mosley', 'NNP'), ('as', 'IN'), ('follows',
'VBZ'), (':', ':'), ('(', '('), ('1', 'CD'), (')',
')'), ('payment', 'NN'), ('for', 'IN'), ('the', 'DT'),
('concrete', 'NN'), ('slab', 'NN'), ('upon', 'IN'),
('completion', 'NN'), ('of', 'IN'), ('all', 'DT'),
('slab', 'NN'), ('and', 'CC'), ('foundation', 'NN'),
('work', 'NN'), (',', ','), ('including', 'VBG'),
('authorized', 'JJ'), ('additions', 'NNS'), (';',
':'), ('(', '('), ('2', 'CD'), (')', ')'), ('payment',
'NN'), ('for', 'IN'), ('cost', 'NN'), ('of', 'IN'),
('building', 'NN'), ('materials', 'NNS'), ('upon',
'IN'), ('delivery', 'NN'), ('of', 'IN'), ('the',
'DT'), ('materials', 'NNS'), ('to', 'TO'), ('GCF',
'NNP'), ("'s", 'POS'), ('plant', 'NN'), (';', ':'),
('and', 'CC'), ('(', '('), ('3', 'CD'), (')', ')'),
('payment', 'NN'), ('of', 'IN'), ('the', 'DT'),
('balance', 'NN'), ('of', 'IN'), ('the', 'DT'),
('contract', 'NN'), ('upon', 'IN'), ('completion',
'NN'), ('of', 'IN'), ('the', 'DT'), ('building',
'NN'), ('.', '.')]

  1115-> . The contract price for
  [('The', 'DT'), ('contract', 'NN'), ('price', 'NN'),
('for', 'IN'), ('slab', 'NN'), ('and', 'CC'),
('foundation', 'NN'), ('work', 'NN'), ('was', 'VBD'),
('$', '$'), ('20,714.00', 'CD'), (';', ':'), ('cost',
'NN'), ('of', 'IN'), ('the', 'DT'), ('building',
'NN'), ('materials', 'NNS'), ('was', 'VBD'), ('$',
'$'), ('46,031.00', 'CD'), (',', ','), ('and', 'CC'),
('a', 'DT'), ('balance', 'NN'), ('of', 'IN'), ('$',
'$'), ('8,537.00', 'CD'), ('was', 'VBD'), ('due',
'JJ'), ('upon', 'IN'), ('completion', 'NN'), ('of',
'IN'), ('the', 'DT'), ('building', 'NN'), ('.', '.')]

  1153-> for the contract was $
  [('Total', 'JJ'), ('consideration', 'NN'), ('for',
'IN'), ('the', 'DT'), ('contract', 'NN'), ('was',
'VBD'), ('$', '$'), ('75,282.00', 'CD'), ('.', '.')]
```

Fig. 11. Some retrieved sentences by the function sentence.

These retrieved sentences illustrate the contract between the plaintiff and the defendant. However, some retrieved sentences are related more to the reasoning and facts of the case than to the contract. Some of them are listed as follows:

```
  1651-> . We agree that the
  [('We', 'PRP'), ('agree', 'VBP'), ('that', 'IN'),
('the', 'DT'), ('prefabricated', 'JJ'), ('building',
'NN'), ('is', 'VBZ'), ('a', 'DT'), ('``', '``'),
('good', 'JJ'), ("''", "''"), ('under', 'IN'),
('Article', 'NNP'), ('2', 'CD'), ('of', 'IN'), ('the',
'DT'), ('U.C.C.', 'NNP'), (',', ','), ('but', 'CC'),
('we', 'PRP'), ('do', 'VBP'), ('not', 'RB'), ('agree',
'VB'), ('that', 'IN'), ('the', 'DT'), ('U.C.C',
'NNP'), ('.', '.')]

  1842-> to the contract . HN2
  [('The', 'DT'), ('dispositive', 'JJ'), ('question',
'NN'), (',', ','), ('however', 'RB'), (',', ','),
('is', 'VBZ'), ('not', 'RB'), ('whether', 'IN'),
('the', 'DT'), ('new', 'JJ'), ('building', 'NN'),
('conforms', 'NNS'), ('to', 'TO'), ('the', 'DT'),
('existing', 'VBG'), ('building', 'NN'), (',', ','),
('but', 'CC'), ('whether', 'IN'), ('it', 'PRP'),
('conforms', 'VBZ'), ('to', 'TO'), ('the', 'DT'),
('contract', 'NN'), ('.', '.')]
```

Fig. 12. Some retrieved sentences that are more related to the reasoning and facts of the case.

To differentiate the sentences in Fig. 11 from those in Fig. 12, note that the word "agreed" in Fig. 11 has the tag VBD, meaning that it is a past tense verb; but the word "agree" in Fig. 12 has the tags VBP and VB, meaning that it is respectively a non-third-person singular present tense verb and a base form verb. The contract of the example court decision must have been made in the past. Hence, the word "agreed" in Fig. 11 gives us a clue that the sentences in Fig. 11 are related to the formation of the contract. In addition, "Mosley" and "GCF", the plaintiff and the defendant of the case, are the singular, proper nouns (NNP) in the sentences in Fig. 11. However, "We", a personal pronoun, is the subject of the sentence in Fig. 12. This provides more evidences to our claim. For the word "contract," both sentences in Fig. 11 contain dollar amounts while the one in Fig. 12 does not. This provides us clues to differentiate the sentences in Fig. 11 and 12.

## IV. Conclusions

This paper presents the steps that we take to extract relevant sentences from a case file according to an ontology created for the type of law suits of interest. Although this is only the beginning of our investigation, this research project has already shown promises in developing automated systems to facilitate legal researches, lighting the burdens of legal practitioners.

Many obstacles remain to be overcome. Although the NLTK module is constantly making progress, some of its functions need further improvements. Note that one of the sentences in Fig. 12 is not complete no matter whether our own function sentence or the one provided by NLTK was used. Hence, tokenization must be done better. In addition, after relevant passages are extracted, more concrete information from the extracted sentences need to be garnered before we can feed those information into a deep learning system.

## References


[1] S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python*, O'Reilly Media Inc. 2009.

[2] K. Clark, D. Sharma, R. Qin, C. G. Chute, and C. Tao, "A use case study on late stent thrombosis for ontology-based temporal reasoning and analysis," *Journal of Biomedical Semantics*, vol. 5, no. 1, pp. 49, 2014.

[3] B. Fazzinga, G. Gianforme, G. Gottlob, and T. Lukasiewicz, "Semantic web search based on ontological conjunctive queries," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 4, pp. 453-473, 2011

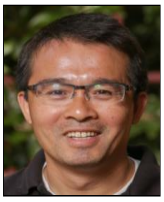
[4] A. Kementsietsidis, L. Lim, and M. Wang, "Supporting ontology-based keyword search over medical databases," *AMIA Annual Symposium Proceedings*, pp. 409-413, 2008.

[5] D. Lonsdale, D. Embley, Y. Ding, L. Xu, and M. Hepp, "Reusing ontologies and language components for ontology generation," *Data & Knowledge Engineering*, vol. 69, no. 4, pp. 318-330, 2010.

[6] M. Silberstein and J. McGeever, "The search for ontological emergence," *The Philosophical Quarterly*, vol. 49, no. 195, pp. 182-200, 1999.

[7] C. Tao and D. Embley, "Automatic hidden-web table interpretation, conceptualization, and semantic annotation," *Data & Knowledge Engineering*, vol. 68, no. 7, pp. 683-703, 2009.

[8] N. Xue, "Book review: S. Bird, E. Loper, and E. Klein, natural language processing with python," *Natural Language Engineering*, vol. 17, no. 3, pp. 419-424, 2011.

**Wai Yin Mok** is a professor of information systems at the University of Alabama in Huntsville. His research interests include database systems, business workflows, natural language processing and machine learning. He has published papers in ACM Transactions of Database Systems, IEEE Transactions of Knowledge and Data Engineering, Information Systems, Decision Support Systems, Data and Knowledge Engineering and many other journals.