South Korean Legal Document Classification Using LSTM

Roy GeonWoo Park

DRP Final Draft, September 30, 2019

<Abstract>

Studies have tested the effectiveness of Long-Short Term Memory (LSTM) for classification of legal documents in the United States. This study tests its effectiveness for South Korean legal document classification, using 14,526 compensation records and 9,271 tax law precedent records collected from the library of the Supreme Court of South Korea. The documents were preprocessed using the KoNLP Okt Korean morpheme analyzer and classified using a sequential model constructed with the Keras Python deep learning library. The LSTM sequential model provided 60% accuracy, but may be able to improve on this result with more refined data. This suggests that LSTM may be useful for document classification and information retrieval in the South Korean legal system.

Introduction

The application of artificial intelligence in the legal industry boostered the legal tech field. There already are commercialized artificial intelligence lawyers such as DoNotPay, Ross, and i-LIS, which assist lawyers by reducing time spent searching for case-relevant information.[1] The performance of i-LIS from a legal advising competition that surpassed human lawyers reflects the power of artificial intelligence lawyers.[2]

This paper explores the effectiveness of LSTM's application on Korean legal document classification. This paper demonstrates the potential of LSTM's use as a legal document classification model based on a binary document classification of Korean legal documents.

The development of software that automates the process of collecting case-relevant information is in the field of text mining. Text mining refers to a process of extracting useful information from text data. Major text mining fields are document classification, clustering, information retrieval, information extraction, and sentiment analysis.[3] Among those, document classification and information retrieval are core technologies of the legal tech field. The task of document classification is to categorize documents based on their content or users' request while information retrieval aims to return a list of sorted documents based on the documents' relevance to users' input queries.

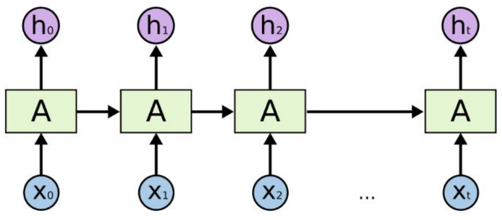
Both document classification and information retrieval are processed through deep learning. Deep learning is a type of machine learning that allows processing of unstructured data, which includes text data, pictures, and videos. Deep learning networks consist of artificial neural networks that differentiate deep learning from machine learning. Artificial neural networks consist of several hidden layers that enable deep learning networks to realize underlying relationships in a set of data by themselves.

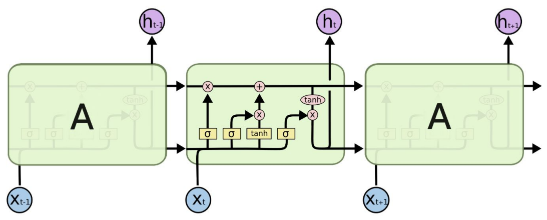
[1] Hye-Ri An, "[논설위원이 간다]로펌 간 한국 첫 AI 변호사…검사도 놓친 분석 '단 20초'," *The JoongAng Ilbo*, April 5, 2018, https://news.joins.com/article/22508494.

[2] Eun-Ju Bang, "AI변호사 1~3위, 사람 변호사들 완패...참여자들 'AI능력 놀라워'," *ZD Net Korea*, August 29, 2019, https://www.zdnet.co.kr/view/?no=20190829232940.

[3] Sholom M. Weiss et al, "Looking for Information in Documents." *Texts in Computer Science Fundamentals of Predictive Text Mining*, (2010).

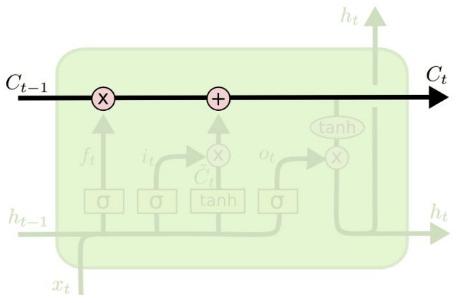
Two big branches of artificial neural networks are convolutional neural networks (CNN) and recurrent neural networks (RNN). CNN models consist of convolution layers, pooling layers, and fully connected layers. The models are considered efficient for semantic parsing and processing 2-dimensional data.[4] The models could be trained by a backpropagation algorithm.[5] RNN models consist of several loops that allow their trained data to persist.[6] However, the models have a problem of long-term dependencies which restricts the size of the data the models can process.[7]





<Figure 1>[8] <Figure 2>[9]

Long Short-Term Memory (LSTM) is a type of RNN that compensates the defect of RNN. LSTM models have a cell state that allows the models to add or remove trained data through gates. Such a system allows the models to have more sophisticated control over the long-term dependencies problem of the RNN models by determining the amount of memory loss through each gate.[10]



<Figure 3>[11]

CNN and LSTM are often used for document classification problems because of their suitability for natural language processing. Comparative Study of CNN and RNN for Natural

[4] Wen-tau Yih et al, "Semantic Parsing for Single-Relation Question Answering," *Microsoft Research WA* 98052.

[5] Yann LeCun et al, "gradient-based learning applied to Document Recognition," *PROC. OF THE IEEE*, (1998).

[6] Jeffrey L. Elman, "Finding Structure in Time," *Cognitive Science* 14, (1990): 179-211.

[7] Yoshua Bengio et al, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions On Neural Networks* 5, no. 2 (1994).

[8] Christopher Olah, *An Unrolled Recurrent Neural Network*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[9] Christopher Olah, *An Unrolled Recurrent Neural Network*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[10] Sepp Hochreiter et al, "LONG SHORT-TERM MEMORY," *Neural Computation* 9, no. 8 (1997): 1735-1780.

[11] Christopher Olah, *An Unrolled Recurrent Neural Network*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Language Processing[12] does a systematic comparison of the two models on a wide range of representative natural language problems. The paper displays that the CNN models work better for relation classification[13] whereas the LSTM models are more effective for sentiment classification[14]. A Comparative Study of Classifying Legal Documents with Neural Networks [15] specifically examines the best fitting artificial neural network model for the field of legal document classification.

While extensive amount of studies about the applications of RNN and LSTM to document classification problems is done, there is still room for more studies. Considering that the studies are done with documents written in English, conducting similar experiments with different languages is worthwhile as distinctive linguistic characteristics of the languages may affect the result. Korean is not an exception. Korean is a language with more parts of speech than English. Sentence structures are also different in Korean.[16]

Studies of Korean document classification using CNN and LSTM has made much progress. Cho etl demonstrated CNN's adequacy on Korean document classification problems.[17] Lee approached Korean document classification problems with RNN.[18] Lee and Cho found a best combination of RNN and CNN for the most efficient model for Korean document classification problems.[19] However, not much research is done with Korean legal documents. Two studies are done in the field of Korean legal document classification. A Study on Legal Document Classification Systems[20] follows the procedure of Yoon Kim's paper[21] and shows 65.53% accuracy. As an extension of A Study on Legal Document Classification Systems, A Study on Legal Document Classification Systems Using TF-IDF and CNN[22] combines TF-IDF with CNN and succeeds raising the accuracy to 81.08%. However, no Korean legal document classification study has been done with RNN yet.

Method

[12] Wenpeng Yin et al, "Comparative Study of CNN and RNN for Natural Language Processing," *IBM Research,* (2017).

[13] Iris Hendrickx et al, "SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals"

[14] Richard Socher at el, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank"

[15] Samir Undavia and John Ortega, "A Comparative Study of Classifying Legal Documents with Neural Networks," *ResearchGate* 10, (2018)

[16] Jung-Song Jae, *THE KOREAN LANGUAGE* (New York: Routledge, 2005).

[17] John Kim et al, "Large-Scale Text Classification Methodology with Convolutional Neural Network," 동계학술발표회 논문집, (2015).

[18] John Kim et al, "Large-Scale Text Classification with Recurrent Neural Networks," 한국컴퓨터종합학술대회 논문집, (2016).

[19] Yeong-Su Kim and Seung-Woo Lee, "Combinations of Text Preprocessing and Word Embedding Suitable for Neural Network Models for Document Classification," *Journal of KIISE* 45, no. 7 (2018): 690-700.

[20] Ji-Hoon Lee et al, "A Study on Legal Document Classification Systems," *Korea Institute of Communication Sciences*, (2018): 939-940.

[21] Yoon Kim, "Convolutional Neural Networks for Sentence Classification"

[22] Ji-Hoon Lee et al, "A Study on Legal Document Classification Systems Using TF-IDF and CNNs," *Korea Institute of Communication Sciences*, (2019): 982-983.

BACKGROUND

A Keras sequential model is used to categorize legal documents into either compensation or taxation cases. Keras is a python deep learning library built upon Tensorflow, CNTK, and Theano deep learning. I decided to use Keras because it offers simple yet effective functions for LSTM than the other deep learning frameworks.

The goal of this experiment is to gather as many legal documents as possible and automate the process of categorization without human interaction. The following is the major steps of the experiment. First, 14,526 case documents of compensation law and 9,271 tax law are collected from the official online library of the Supreme Court of Korea. The documents include any civil, criminal, administrative, patent, family, and tax law precedents from Supreme Courts, High Courts, and Lower Courts that fall into the category of compensation law and tax law. Each case document consists of Holding, Headnote, Relevant Provision, Relevant Precedents, and the Full Text sections. When collected, compensation law documents are labeled '0' and tax law documents are labeled '1'. Then, the documents are divided into a training data set and a testing data set. 80% of the compensation and tax documents are randomly selected for the training data set for training a Keras sequential model and the rest of the documents are integrated into the testing data set for testing the accuracy of the training. At the same time, both of the text data sets are preprocessed. The preprocessing consists of tokenization, normalization, and elimination of stopwords. KoNLPy's Okt Korean morpheme analyzer is used for data pre-processing because it has faster processing speed than the other morpheme analyzers such as KKMA and MeCab-ko. Next, the preprocessed data sets are embedded using a word2vec method. Finally, the embedded training data set is used for training a Keras sequential model and the level of achievement of the training is tested with the testing data set.

PREPARATION FOR TRAINING KERAS

Before utilizing Keras for performing document classification, 3 steps of preparation are done: Standardizing the data to fit the Keras specifications, constructing a sequential model to process the standardized data, and configuring the learning process of Keras.

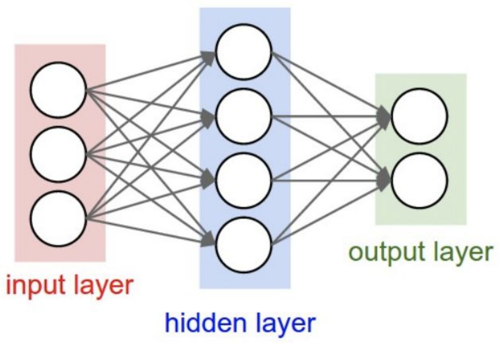
The core of standardizing the data is preprocessing. There are various ways to do data preprocessing yet this paper employs tokenization, normalization, and elimination of stopwords processes as these are relatively important steps of data preprocessing.

Tokenization is a process of chopping a text document into a sequence of tokens. English tokenization is relatively simple because tokens are classified based on spacing words except for a few cases like contractions such as "don't" or phrases such as "New York." However, in case of Korean, often, words combined with postposition particles without spacing and words with a root and an ending unified make tokenization tricky. The subjective rule of spacing words for compound words also hinders easy Korean tokenization. KoNLPy's Okt library is used

Normalization is a process of simplifying the transformed tokens into the basic form of the tokens. Such process enhances the work efficiency of text mining by reducing the number of the types of tokens. On the contrary, the risk of some degree of semantic loss follows. For instance, “love” and “loved” can be normalized into the fundamental token “love” yet the meaning of tense difference disappears. Normalization is relatively challenging for languages like English and German. Including the verb form variation, these languages have many aspects to be considered when normalized. For instance, the first letter of the first word of a sentence must be written in a capital letter. Meanwhile, Korean normalization is relatively easy because the language does not have much form variations.

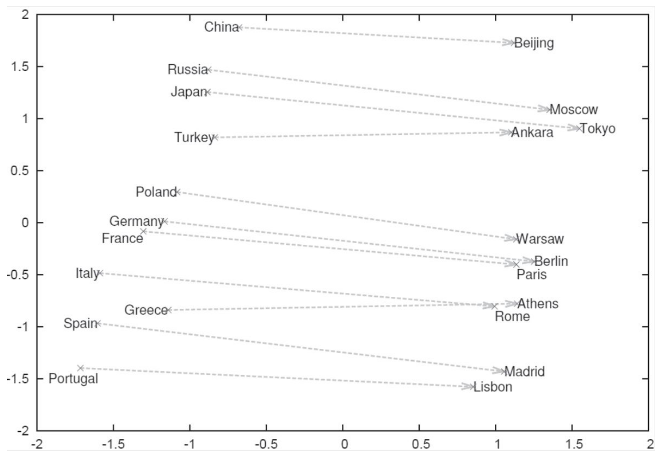
Elimination of stopwords is a process of getting rid of insignificant words with little semantic value. Getting rid of such stopwords increases the level of achievement of a Keras sequential model training. 30 stopwords are eliminated based on frequency.

After preprocessing, Keras is used to conduct integer encoding. Encoding is a process of transforming text data into a numerical form. While there are many types of encoding, Keras requires integer encoding for the word embedding process. After the integer encoding process, the collected data sets are standardized to fit the Keras specifications.



<Figure 4>

Then, a Keras sequential model is constructed. Figure 4 represents a model of a sequential model. In this experiment, the model consists of 3 layers: A word embedding layer, a LSTM layer, and a sigmoid layer.



<Figure 5>[23]

Word embedding is a process of transforming the encoded numerical data into a vector form. It is based on a statistical semantics hypothesis that “statistical patterns of

[23] Won Son, “Word2Vec,” 2018.

human word usage can be used to figure out what people mean"[24]. This means that a combination of words in similar contexts will have similar meanings. Based on this hypothesis, insignificant details of sentences are not important. Therefore, preprocessing is essential before word embedding. A word2vec method is used for embedding the data into a 100 dimensional vectors as it is renowned for its high accuracy.[25] The word2vec method transforms words with a high semantic similarity into similar word vectors. As a result, an arithmetic operation of the word vectors is possible. Figure 5 is an example of a set of countries and a set of the countries' capitals represented in a vector form using the word2vec method. The set of country vectors and the set of capital vectors are clearly divided. The graph reflects the semantic similarity of each data set.

After both the sets of training and testing data are embedded, the LSTM layer takes the data sets and self-studies to properly categorize documents based on hidden layers. With the 100 dimensional input data, after the self-studying process, the LSTM layer gives 128 dimensional vector data as an output.

Finally, a Sigmoid layer provides classification of tax law and compensation based on the LSTM's output. The sigmoid layer is optimized for binary document classification.

After the Keras sequential model is constructed, the configuration process of the model is taken. Keras has a built-in configuration process using an optimizer (defines the learning rate), a loss function (reduces errors), and a list of metrics (judge the performance of the deep learning). A combination of a "rmsprop" optimizer, a "binary_crossentropy" function, and an "acc" matrix is the best binary classification environment Keras provides.

TRAINING KERAS

With the model fully developed, Keras is now trained to classify documents. Keras requires 4 inputs for the training: training data sets, epochs, cooldown periods to ensure machinery is not overloaded, and the amount of portion of the data sets for a progress inspection. Epochs refers to a frequency of the training iterations using the same data sets. It is important to train the data sets with varying epochs values to prevent the overfitting problem.[26] Therefore, the experiment is repeated with different epochs values to get the highest possible accuracy.

Results

Epochs	Accuracy
3	0.5701
4	0.6026

[24] P. D. Turney et al, "From Frequency to Meaning: Vector Space Models of Semantics." *Journal of Artificial Intelligence Research* 37, (2010): 141-88.

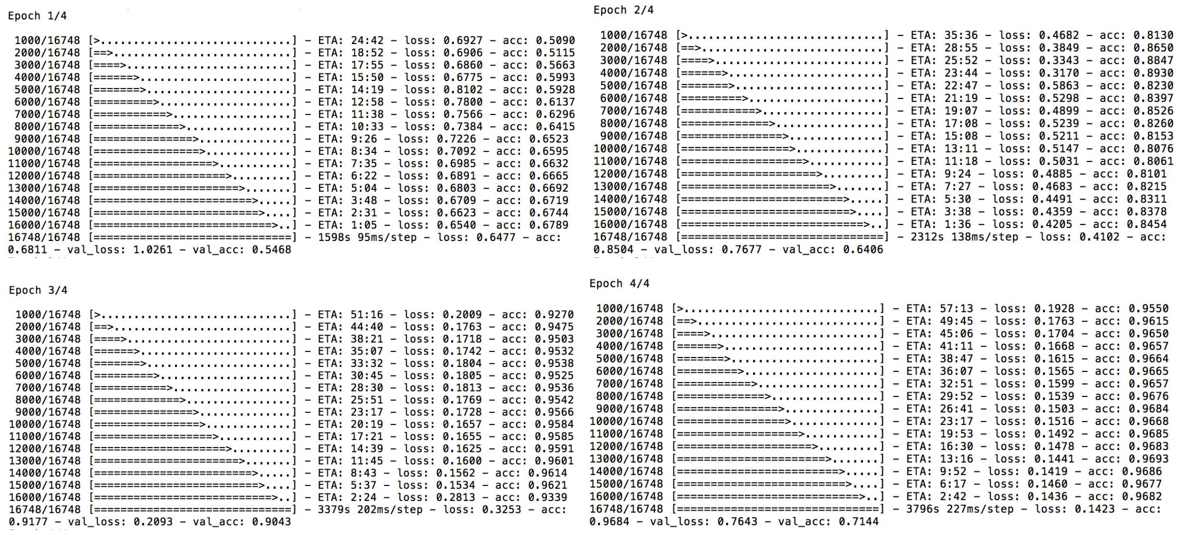
[25] T. Mikolov, "Distributed representations of words and phrases and their compositionality." *In Advances in neural information processing systems*, (2013): 3111-3119.

[26] Wojciech Zaremba et al, "RECURRENT NEURAL NETWORK REGULARIZATION," *ICLR*, (2015).

5	0.5540
6	0.4187

<Figure 6>

The overall accuracy depended on the varying values of epochs. Figure 6 shows the result of the experiments depending on the varying values of epochs. The sequential model gave the highest accuracy of 60% with 4 epochs.



```
Epoch 1/4

 1000/16748 [>.............................] - ETA: 24:42 - loss: 0.6927 - acc: 0.5090
 2000/16748 [==>...........................] - ETA: 18:52 - loss: 0.6906 - acc: 0.5115
 3000/16748 [====>.........................] - ETA: 17:55 - loss: 0.6860 - acc: 0.5663
 4000/16748 [======>.......................] - ETA: 15:50 - loss: 0.6775 - acc: 0.5993
 5000/16748 [=======>......................] - ETA: 14:19 - loss: 0.8102 - acc: 0.5928
 6000/16748 [=========>....................] - ETA: 12:58 - loss: 0.7800 - acc: 0.6137
 7000/16748 [===========>..................] - ETA: 11:38 - loss: 0.7566 - acc: 0.6296
 8000/16748 [=============>................] - ETA: 10:33 - loss: 0.7384 - acc: 0.6415
 9000/16748 [===============>..............] - ETA: 9:26 - loss: 0.7226 - acc: 0.6523
10000/16748 [================>.............] - ETA: 8:34 - loss: 0.7092 - acc: 0.6595
11000/16748 [==================>...........] - ETA: 7:35 - loss: 0.6985 - acc: 0.6632
12000/16748 [====================>.........] - ETA: 6:22 - loss: 0.6891 - acc: 0.6665
13000/16748 [======================>.......] - ETA: 5:04 - loss: 0.6803 - acc: 0.6692
14000/16748 [========================>.....] - ETA: 3:48 - loss: 0.6709 - acc: 0.6719
15000/16748 [=========================>....] - ETA: 2:31 - loss: 0.6623 - acc: 0.6744
16000/16748 [===========================>..] - ETA: 1:05 - loss: 0.6540 - acc: 0.6789
16748/16748 [==============================] - 1598s 95ms/step - loss: 0.6477 - acc:
0.6811 - val_loss: 1.0261 - val_acc: 0.5468
```

```
Epoch 2/4

 1000/16748 [>.............................] - ETA: 35:36 - loss: 0.4682 - acc: 0.8130
 2000/16748 [==>...........................] - ETA: 28:55 - loss: 0.3849 - acc: 0.8650
 3000/16748 [====>.........................] - ETA: 25:52 - loss: 0.3343 - acc: 0.8847
 4000/16748 [======>.......................] - ETA: 23:44 - loss: 0.3170 - acc: 0.8930
 5000/16748 [=======>......................] - ETA: 22:47 - loss: 0.5863 - acc: 0.8230
 6000/16748 [=========>....................] - ETA: 21:19 - loss: 0.5298 - acc: 0.8397
 7000/16748 [===========>..................] - ETA: 19:07 - loss: 0.4899 - acc: 0.8526
 8000/16748 [=============>................] - ETA: 17:08 - loss: 0.5239 - acc: 0.8260
 9000/16748 [===============>..............] - ETA: 15:08 - loss: 0.5211 - acc: 0.8153
10000/16748 [================>.............] - ETA: 13:11 - loss: 0.5147 - acc: 0.8076
11000/16748 [==================>...........] - ETA: 11:18 - loss: 0.5031 - acc: 0.8061
12000/16748 [====================>.........] - ETA: 9:24 - loss: 0.4885 - acc: 0.8101
13000/16748 [======================>.......] - ETA: 7:27 - loss: 0.4683 - acc: 0.8215
14000/16748 [========================>.....] - ETA: 5:30 - loss: 0.4491 - acc: 0.8311
15000/16748 [=========================>....] - ETA: 3:38 - loss: 0.4359 - acc: 0.8378
16000/16748 [===========================>..] - ETA: 1:36 - loss: 0.4205 - acc: 0.8454
16748/16748 [==============================] - 2312s 138ms/step - loss: 0.4102 - acc:
0.8504 - val_loss: 0.7677 - val_acc: 0.6406
```

```
Epoch 3/4

 1000/16748 [>.............................] - ETA: 51:16 - loss: 0.2009 - acc: 0.9270
 2000/16748 [==>...........................] - ETA: 44:40 - loss: 0.1763 - acc: 0.9475
 3000/16748 [====>.........................] - ETA: 38:21 - loss: 0.1718 - acc: 0.9503
 4000/16748 [======>.......................] - ETA: 35:07 - loss: 0.1742 - acc: 0.9532
 5000/16748 [=======>......................] - ETA: 33:32 - loss: 0.1804 - acc: 0.9538
 6000/16748 [=========>....................] - ETA: 30:45 - loss: 0.1805 - acc: 0.9525
 7000/16748 [===========>..................] - ETA: 28:30 - loss: 0.1813 - acc: 0.9536
 8000/16748 [=============>................] - ETA: 25:51 - loss: 0.1769 - acc: 0.9542
 9000/16748 [===============>..............] - ETA: 23:17 - loss: 0.1728 - acc: 0.9566
10000/16748 [================>.............] - ETA: 20:19 - loss: 0.1657 - acc: 0.9584
11000/16748 [==================>...........] - ETA: 17:21 - loss: 0.1655 - acc: 0.9585
12000/16748 [====================>.........] - ETA: 14:39 - loss: 0.1625 - acc: 0.9591
13000/16748 [======================>.......] - ETA: 11:45 - loss: 0.1600 - acc: 0.9601
14000/16748 [========================>.....] - ETA: 8:43 - loss: 0.1562 - acc: 0.9614
15000/16748 [=========================>....] - ETA: 5:37 - loss: 0.1534 - acc: 0.9621
16000/16748 [===========================>..] - ETA: 2:24 - loss: 0.2813 - acc: 0.9339
16748/16748 [==============================] - 3379s 202ms/step - loss: 0.3253 - acc:
0.9177 - val_loss: 0.2093 - val_acc: 0.9043
```

```
Epoch 4/4

 1000/16748 [>.............................] - ETA: 57:13 - loss: 0.1928 - acc: 0.9550
 2000/16748 [==>...........................] - ETA: 49:45 - loss: 0.1763 - acc: 0.9615
 3000/16748 [====>.........................] - ETA: 45:06 - loss: 0.1704 - acc: 0.9650
 4000/16748 [======>.......................] - ETA: 41:11 - loss: 0.1668 - acc: 0.9657
 5000/16748 [=======>......................] - ETA: 38:47 - loss: 0.1615 - acc: 0.9664
 6000/16748 [=========>....................] - ETA: 36:07 - loss: 0.1565 - acc: 0.9665
 7000/16748 [===========>..................] - ETA: 32:51 - loss: 0.1599 - acc: 0.9657
 8000/16748 [=============>................] - ETA: 29:52 - loss: 0.1539 - acc: 0.9676
 9000/16748 [===============>..............] - ETA: 26:41 - loss: 0.1503 - acc: 0.9684
10000/16748 [================>.............] - ETA: 23:17 - loss: 0.1516 - acc: 0.9668
11000/16748 [==================>...........] - ETA: 19:53 - loss: 0.1492 - acc: 0.9685
12000/16748 [====================>.........] - ETA: 16:30 - loss: 0.1478 - acc: 0.9683
13000/16748 [======================>.......] - ETA: 13:16 - loss: 0.1441 - acc: 0.9693
14000/16748 [========================>.....] - ETA: 9:52 - loss: 0.1419 - acc: 0.9686
15000/16748 [=========================>....] - ETA: 6:17 - loss: 0.1460 - acc: 0.9677
16000/16748 [===========================>..] - ETA: 2:42 - loss: 0.1436 - acc: 0.9682
16748/16748 [==============================] - 3796s 227ms/step - loss: 0.1423 - acc:
0.9684 - val_loss: 0.7643 - val_acc: 0.7144
```

<Figure 7>

Figure 7 shows the process of the sequential model's training. During the first epoch, the accuracy started from about 50% and the loss indicated was about 70%. However, as the model was trained with the training data sets, at the end of the fourth epoch, the accuracy rised up to 96% and the loss was lowered up to 14%.

Discussion

Compared to the accuracy of the document classification done with CNN and TF-IDF from the paper by Ji-Hoon Lee et al, the LSTM sequential showed a lower accuracy. However, it is difficult to tell that CNN is more suitable for legal document classification because this experiment had different data sets processed with a different deep learning library from the Ji-Hoon Lee et al's.

Furthermore, experiments done under different conditions have the possibility of increasing the accuracy. In other words, there is room for improvement. For instance, trying a trial with different values of vector dimensions or better preprocessed data may give a better result. The limitations of time and equipments did not allow me to try many trials to examine the most optimized condition for the LSTM sequential model.

Also, the fact that the data sets included every section of the collected legal documents may have decreased the accuracy of the training. Each document consisted of Holding, Headnote, Relevant Provision, Relevant Precedents, and the Full Text sections and each section had different role. Therefore, using all the sections may have increased the inconsistency of the data.

Although this paper focuses on examining the capability of LSTM for legal document classification with binary document classification, this paper may be utilized as a foundation for a larger comparison research of CNN and RNN application for Korean legal document classification. Moreover, incorporating more groups of data into this experiment would enable this experiment to handle both document classification problems and information retrieval problems.

Finally, I would like to deliver special thanks to Seok-Min Kim, Dr. Jennings, and Ms. Chang for assisting me to finish this DRP.

Bibliography

An, Hye-Ri. “[논설위원이 간다]로펌 간 한국 첫 AI 변호사…검사도 놓친 분석 '단 20초'” ([Here
Comes The Cirtique]An AI Lawyer Excels in A Law Firm), *The JoongAng Ilbo*, April 5, 2018, https://news.joins.com/article/22508494.

Bang, Eun-Ju. “AI변호사 1~3위, 사람 변호사들 완패...참여자들 ‘AI능력 놀라워’” (AI Lawyer
Surpasses Human Lawyers), *ZD Net Korea*, August 29, 2019, https://www.zdnet.co.kr/view/?no=20190829232940.

Bengio, Yoshua et al. “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions On Neural Networks* 5, no. 2 (1994).

Elman, Jeffrey L.. “Finding Structure in Time,” *Cognitive Science* 14, (1990): 179-211.

Hochreiter, Sepp et al. “LONG SHORT-TERM MEMORY,” *Neural Computation* 9, no. 8 (1997): 1735-1780.

Hendrickx, Iris et al. “SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations
Between Pairs of Nominals”

Jae, Jung-Song. *THE KOREAN LANGUAGE* (New York: Routledge, 2005).

Kim, John et al. “Large-Scale Text Classification Methodology with Convolutional Neural Network,” 동계학술발표회 논문집 (Winter Academics Conference), (2015).

Kim John et al. “Large-Scale Text Classification with Recurrent Neural Networks,”
한국컴퓨터종합학술대회 논문집 (Korea Computer Academia Competition),
(2016).

Kim, Yeong-Su and Lee Seung-Woo. “Combinations of Text Preprocessing and Word Embedding Suitable for Neural Network Models for Document Classification,” *Journal of KIISE* 45, no. 7 (2018): 690-700.

Kim, Yoon. “Convolutional Neural Networks for Sentence Classification”

LeCun, Yann et al. “gradient-based learning applied to Document Recognition,” *PROC. OF THE IEEE*, (1998).

Lee, Ji-Hoon et al. “A Study on Legal Document Classification Systems,” *Korea Institute of Communication Sciences*, (2018): 939-940.

Lee, Ji-Hoon et al. “A Study on Legal Document Classification Systems Using TF-IDF and CNNs,” *Korea Institute of Communication Sciences*, (2019): 982-983.

Mikolov, T.. “Distributed representations of words and phrases and their compositionality.” *In*
Advances in neural information processing systems, (2013): 3111-3119.

Olah, Christopher. *An Unrolled Recurrent Neural Network*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Richard, Socher at el. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”

Son, Won. “Word2Vec,” 2018.

Turney, P. D. et al. "From Frequency to Meaning: Vector Space Models of Semantics." *Journal of Artificial Intelligence Research* 37, (2010): 141-88.

Undavia, Samir and Ortega, John. “A Comparative Study of Classifying Legal Documents with Neural Networks,” *ResearchGate* 10, (2018)

Weiss, Sholom M. et al. "Looking for Information in Documents." *Texts in Computer Science Fundamentals of Predictive Text Mining*, (2010).

Yih, Wen-tau et al. “Semantic Parsing for Single-Relation Question Answering,” *Microsoft Research WA* 98052.

Yin, Wenpeng et al. “Comparative Study of CNN and RNN for Natural Language Processing,” *IBM Research,* (2017).

Zaremba, Wojciech et al. “RECURRENT NEURAL NETWORK REGULARIZATION,” *ICLR*, (2015).