Automatic Catchphrase Extraction from Legal Case Documents via Scoring using Deep Neural Networks

Vu Tran*, Minh Le Nguyen*, and Ken Satoh†

*Japan Advanced Institute of Science and Technology
†National Institute of Informatics, Japan
vu.tran@jaist.ac.jp, nguyenml@jaist.ac.jp, ksatoh@nii.ac.jp

Abstract. In this paper, we present a method of automatic catchphrase extracting from legal case documents. We utilize deep neural networks for constructing scoring model of our extraction system. We achieve comparable performance with systems using corpus-wide and citation information which we do not use in our system.

Keywords: catchphrase generation, legal case documents, convolutional neural networks

1 Introduction

The growth of legal documents overtime and their absurd length raise the need of automatic legal document processing systems. One of the processing steps is to identify the gist of the documents, specifically, catchphrases for legal case documents. “Catchphrases have an indicative function rather than informative, they present all the legal point considered instead that just summarizing the key points of a decision” [3]. Catchphrases give a quick impression on what the case is about: “the function of catchwords is to give a summary classification of the matters dealt with in a case. [...] Their purpose is to tell the researcher whether there is likely to be anything in the case relevant to the research topic” [11]. On one hand, catchphrases help lawyers/researchers quickly grasp the points of a case, without having to read the entire document, which saves significant time and effort for finding/studying relevant cases. On the other hand, catchphrases help improves the performance of automatic case retrieval systems.

Despite of the benefits, catchphrases are not always available in legal case documents, and are drafted by legal experts, which requires huge efforts when considering the enormous number of legal case documents. It is, therefore, crucial to build automatic catchphrase generation systems for both old documents not having drafted catchphrases and new documents. Developing such systems, however, is challenging as the complexity of catchphrases shown in Table 1.

Approaches for generating catchphrases are based on phrase scoring derived from common model for retrieval: lexical matching with term frequency-inverse document frequency [2,3,9]. The approaches are bounded by the limit of lexical matching, and corpus-wide statistical information. The limit of lexical matching
can be lifted by moving to distributed vector space, for instance, distributed word embeddings in which common models are Word2Vec \[10\] and GloVe \[12\]. Corpus-wide statistical information has limit capability to identify catchphrases which are not really specific to some document but commonly used in several others.

We propose to build a learning model to extract catchphrases for new documents with the knowledge from previously seen documents and the expert drafted catchphrases thereof. Our system utilizes deep neural networks which have been widely used in natural language processing \[8\] to learn the direct relationship between gold catchphrases and document phrases.

**Table 1.** Example of catchphrases found in legal case reports.

| COSTS - proper approach to admiralty and commercial litigation - goods transported under bill of lading incorporating Himalaya clause - shipper and consignee sued ship owner and stevedore for damage to cargo - stevedore successful in obtaining consent orders on motion dismissing proceedings against it based on Himalaya clause - stevedore not furnishing critical evidence or information until after motion filed - whether stevedore should have its costs - importance of parties cooperating to identify the real issues in dispute - duty to resolve uncontentious issues at an early stage of litigation - stevedore awarded 75% of its costs of the proceedings |
| CORPORATIONS - winding up - court-appointed liquidators - entry into agreement - able to subsist more than three months - no prior approval under s 477(2B) of Corporations Act 2001 (Cth) - application to extend “period” for approval under s 1322(4)(d) - no relevant period - s 1322(4)(d) not applicable - power of Court under s 479(3) to direct liquidator - liquidator directed to act on agreement as though approved - implied incidental powers of Court - prior to approve agreement - power under s 1322(4)(a) to declare entry into agreement and agreement not invalid - COURTS AND JUDGES - Federal Court - implied incidental power - inherent jurisdiction |
| MIGRATION - partner visa - appellant sought to prove domestic violence by the provision of statutory declarations made under State legislation - “statutory declaration” defined by the Migration Regulations 1994 (Cth) to mean a declaration “under” the Statutory Declarations Act 1959 (Cth) in Div 1.5 - contrary intention in reg 1.21 as to the inclusion of State declarations under s 27 of the Acts Interpretation Act - statutory declaration made under State legislation is not a statutory declaration “under” the Commonwealth Act - appeal dismissed |

2 Related Work

In the task of automatic catchphrase generation of legal case documents, various types of information are used in building the generation systems. In \[3\], they use multiple phrase statistics from sentence-wide to corpus-wide for phrase scoring, which are derived from term frequency and inverse document frequency. Latter, in \[2\], they use the citation information as additional information source. Both \[2,3\], however, focus on building sentence extraction systems where the
systems should produce sentences containing catchphrases. In the work of [9], they approach the task as exact catchphrase identification. Their system extracts phrases using word n-grams (n=1,2,...,7), then scoring the phrases also by phrase statistics. Furthermore, they use a large dictionary of legal terms as additional source. Our system, however, only uses document sentences and gold drafted catchphrases as learning resources.

3 Our Approach

Our catchphrase extraction system has two phases: Scoring and Selection. Firstly, we train a scoring model using deep learning networks with our proposed objectives for optimizing the model. The optimization task is similar to a learning to rank task where we rank catchphrases higher than normal sentence phrases. Secondly, we select phrases having highest anchor scores as output catchphrases.

3.1 Phase 1: Scoring

In this phase, we present our scoring model and how to train it using documents and their corresponding drafted catchphrases.

Constructing our scoring model architecture. We score each word in a document based on its contexts: neighbor words, enclosing sentence, and document.

We adapt convolutional neural networks (CNNs), which are successfully used in text modeling [6,13,4,5], to encode each word with its surrounding words into latent feature space, namely word feature vector which represents how the word is used locally. Specifically, sentence word (catchword) features are captured by applying convolutional operations with window size 2k + 1 covering the word, k left and k right neighbors.

Given a document, we denote \( w_{ji} \) (\( w_{ci} \)), word \( j^{th} \) of sentence (catchphrase) \( i^{th} \). The features of word \( w_j \) of some sentence (catchphrase) are captured using CNNs as follows:

\[
f_{wj} = ReLU \left( W^c \begin{bmatrix} v(w_{j-k}) \\ v(w_{j}) \\ \vdots \\ v(w_{j+k}) \end{bmatrix} \right)
\]

(1)

where, \( v(\cdot) : \rightarrow \mathbb{R}^d \): word embedding vector lookup map, \( \cdot \) \( \in \mathbb{R}^{dk} \): concatenated embedding vector, \( W^c \in \mathbb{R}^{c \times dk} \): convolution kernel matrix with \( c \) filters, \( f_{wj} \in \mathbb{R}^c \): word feature vector, \( ReLU \): rectified linear unit activation.

1 A catchword is just any word in the containing catchphrase without any special implication.
Sentence (catchphrase) features are, then, captured by applying max pooling over the whole sentence (catchphrase).

\[ f_{si} = \text{max-pooling}_j(f_{w^s_j}) \] (2)

\[ f_{ci} = \text{max-pooling}_j(f_{w^c_j}) \] (3)

where max-pooling are operated over each dimension of vectors \( f_{w^s_j}, f_{w^c_j} \).

Document features are captured by applying max pooling over the document (not including gold catchphrases). With the same max-pooling operation as above, we compute document features as:

\[ f_d = \text{max-pooling}_i(f_{si}) \] (4)

The document features depend on only the document sentence, thereby, independent from the gold catchphrases which are obviously not available for new documents.

Finally, we apply a multilayer perceptron (MLP) with one hidden and one output layer

\[ MLP(x) = \text{sigmoid}(W_2 \cdot \tanh(W_1 \cdot x + b_1) + b_2) \] (5)

to compute the score of each word (catchword) \( w^s_j \) (\( w^c_j \)) as

\[ P(w_s, s, d) = MLP\left( \begin{bmatrix} f_{w^s_j} \\ f_{si} \\ f_d \end{bmatrix} \right) \] (6)

\[ P(w_c, c, d) = MLP\left( \begin{bmatrix} f_{w^c_j} \\ f_{ci} \\ f_d \end{bmatrix} \right) \] (7)

Where the hidden layer computes the word representative features respecting to its local use (surrounding neighbors), its enclosing sentence (phrase), and its document. The word representative features are feed to the output layer to compute word score (ranging from 0.0 to 1.0).
**Fig. 1.** Scoring model pipeline. Components linked with double dotted lines have shared parameters/weights.

**Training our scoring model** Main objective: given a document, each of its catchword is “expected” to have higher score than each of its sentence word.

First, we denote mean $E$ and standard deviation $std$ of word scores $P$ for each document $d$ in the following equations, which we will use to describe our objective as set of constraints, then formulated into loss function to be optimized.
\[ E_c = E[P(w_c, c, d)] \text{ where } w_c \in c, c \in d \] (8)

\[ \text{std}_c = \text{std}[P(w_c, c, d)] \text{ where } w_c \in c, c \in d \] (9)

\[ E_s = E[P(w_s, s, d)] \text{ where } w_s \in s, s \in d \] (10)

\[ \text{std}_s = \text{std}[P(w_s, s, d)] \text{ where } w_s \in s, s \in d \] (11)

\[ E_{c,d'} = E[P(w_{c,d'})] \text{ where } w_c \in c, c \notin d' \] (12)

Where \( w, c, s, d \) stand for word, catchphrase, sentence, document respectively.

The main objective is realized by comparing the mean scores of catchwords and sentence words:

(o1) The mean score of catchwords is higher than the mean score of sentence words: \( E_c > E_s \) with the loss term \( \sum_d \max(m_1 - (E_c - E_s), 0) \).

This constraint however can lead to the case wherein the model scores all catchwords high and all sentence words low, which does not reflect the fact that some catchwords are stopwords or punctuations. We, therefore, implement this negative constraint: given two random documents, a catchword of the first document is “expected” to have lower score than a sentence word of the second document.

(o2) The mean score of catchwords is lower than sentence words when comparing catchphrases and sentences not from the same document: \( E_{c,d'} < E_{s,d'} \) with the loss term \( \sum_d \frac{1}{|d'|} \sum_{d' \neq d} \max(m_2 - (E_{s,d'} - E_{c,d'}), 0) \).

The constraint (o2), however, causes an adverse effect of scoring all catchwords low and all sentence word high, since given one document, there is only one positive sample comparing to various negative samples. To leverage the impact, we add the following constraints:

(o3) The maximum score of catchwords is higher than the maximum score of sentence words. We don’t use hard maximum but the estimation \( E + \text{std} \), whereby the constraint is realized as \( (E_c + \text{std}_c) > (E_s + \text{std}_s) \) with the loss term \( \sum_d \max(m_3 - ((E_c + \text{std}_c) - (E_s + \text{std}_s)), 0) \).

(o4) The minimum score of catchwords is higher than the mean score of sentence words. Again, we don’t use hard minimum but the estimation \( E - \text{std} \), whereby the constraint is realized as \( (E_c - \text{std}_c) > E_s \) with the loss term \( \sum_d \max(m_4 - ((E_c - \text{std}_c) - E_s), 0) \).

We also add the following additional constraint to keep the scores from collapsing, which acts as regularization.
(o5) Scores should not have small variance: \(\text{std}_c \neq 0, \text{std}_s \neq 0\). The loss terms are \(-\sum_d \text{std}_c\) and \(-\sum_d \text{std}_s\).

The loss function, hence, is composed from the loss terms formulated from the constraints (o1-5) as follows.

\[
\mathcal{L} = a_1 \sum_d \max(m_1 - (E_c - E_s), 0) \\
+ a_2 \sum_d \frac{1}{|\{d'\}|} \sum_{d' \neq d} \max(m_2 - (E_{s'} - E_{c,d'}), 0) \\
+ b_1 \sum_d \max(m_3 - ((E_c + \text{std}_c) - (E_s + \text{std}_s)), 0) \\
+ b_2 \sum_d \max(m_4 - ((E_c - \text{std}_c) - E_s), 0) \\
- b_3 \sum_d \text{std}_c - b_4 \sum_d \text{std}_s
\]

(13)

Note that rather imposing hard constraints, we compose the loss function with soft constraints. This means that some constraints may not be strictly satisfied after the training process. However, the violations of such constraints still incur certain losses and benefit the learning process.

3.2 Phase 2: Selection

In this phase, we construct phrases from a document as the output catchphrases using the resultant scores in phase 1. For each document,

- Step 1. Selecting top \(t\) anchors: we select top \(t\) highest scored words in the document;
- Step 2. Selecting phrases: for each anchor, we select \(r\) words left and \(r\) words right of the anchor. In this paper, we set \(r = 2k\), according to CNN window size \(2k + 1\), the selected phrase, therefore, covers all word windows containing the anchor.

4 Experiments

4.1 Experimental Settings

Dataset: Legal case reports: published in [3]. We use the data available in [2] for our experiments. The data contains \(\approx 4000\) Australian legal case from 2006 to 2009 with annotations for automatic summarization and citation analysis. Each document contains its catchphrases, citations sentences, citation catchphrases, citation classes, and its segmented sentences. In this paper, we use the [2] https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports
catchphrases and the sentences without any citation information for training our system.

We conducted experiments on four data settings (2006, 2007, 2008, 2009) wherein each year’s data are used as test set and the other years’ data are used as train set with our system parameters described in Table 2.

| Parameter | Description |
|-----------|-------------|
| **Phase 1** | | |
| Embeddings (vector size \(d\)) | GloVe embeddings \(d = 300\) |
| CNN filters \(c\) | 300 |
| CNN window size \(2k + 1\) | 5 |
| MLP hidden size | 300 |
| Optimizer | Adam[1] |
| Learning rate | 0.0001 |
| Gradient clipping max norm | 5.0 |
| Loss coefficients \((a_1, a_2, b_1, b_2, b_3, b_4)\) | \((1.0, 1.0, 0.5, 0.1, 0.01, 0.02)\) |
| Size of negative set \(|\{d'\}|\) | 2 |
| **Phase 2** | | |
| Top \(t\) anchors | 10 |
| Radius \(r = 2k\) | 4 (according to the CNN window size of 5) |

**Evaluation Metrics** As our system produces phrases instead of sentences, we using ROUGE scores [7] to evaluate our method and compare with results in [2]:

- **ROUGE-1**: the count of the uni-gram recall between candidate and reference summaries;
- **ROUGE-SU**: based on skip bi-grams: a skip bi-gram is any pair of words in their sentence order, allowing for arbitrary gaps. Rouge-SU counts all in-order matching word pairs, plus common uni-grams; and
- **ROUGE-W**: based on common sequences with maximum length, with a reward for consecutive matches.

### 4.2 Experimental Results

**Validation of objective.** Apart from the extraction performance, we also check if the constraints defined in Section 3.1 are satisfied on the test data. As illustrated in Fig. 2, some constraints are satisfied but some are not.

\((o1)\) The scoring model has no idea whether the input features coming from catchphrases or document sentences, however, can recognize and score catchphrases with score expectation higher than document sentences.
The negative constraint causes the high variance of catchword scores but
is not strictly satisfied. Although the constraint is not strictly satisfied, the
score model does not fall into scoring all catchwords higher than sentence
words.

✓ (o3) The maximum of catchword scores are statistically higher than the max-
imum of sentence word scores.

✗ (o4) The minimum of catchword scores are statistically not higher than the
mean of sentence word scores.

✓ (o5) The variances of scores are not close to 0.

Fig. 2. Score statistics (mean and standard deviation) for legal case reports from the
test data of the data setting 2006 by our system. Score mean and standard deviation
are represented by each interval centered by the mean value. Upper arrow markers are
catchword scores, lower arrow makers are sentence word scores.
As shown in Table 3, our system performances are stable over different test sets and comparable to the performance of other systems reported in [2].

Table 3. Performance on Legal Case Reports Dataset. Results from our system are presented in four data settings (2006, 2007, 2008, 2009) corresponding to the used test sets. "Average": the results averaged of the four settings. Results copied from [2] are applied on legal case documents in 2010 which are not available in the dataset used in our experiments.

|          | ROUGE-1 |          | ROUGE-SU6 |          | ROUGE-W-1.2 |
|----------|---------|----------|-----------|----------|-------------|
|          | Pre     | Rec      | Fm        | Pre      | Rec         | Fm          | Pre     | Rec      | Fm        |
| Results  |         |          |           |          |             |             |         |          |           |
| of our   |         |          |           |          |             |             |         |          |           |
| system   |         |          |           |          |             |             |         |          |           |
| 2006     | 0.2148  | 0.3244   | 0.2242    | 0.0611   | 0.1186      | 0.0524      | 0.1373  | 0.1508   | 0.1189    |
| 2007     | 0.2258  | 0.3204   | 0.2321    | 0.0644   | 0.1129      | 0.0539      | 0.1404  | 0.1393   | 0.1182    |
| 2008     | 0.2255  | 0.2906   | 0.2202    | 0.0656   | 0.0996      | 0.0501      | 0.1430  | 0.1285   | 0.1141    |
| 2009     | 0.2584  | 0.2982   | 0.2414    | 0.0830   | 0.0999      | 0.0583      | 0.1591  | 0.1267   | 0.1189    |
| Average  | 0.2311  | 0.3084   | 0.2295    | 0.0685   | 0.1078      | 0.0537      | 0.1450  | 0.1363   | 0.1175    |
| Results  |         |          |           |          |             |             |         |          |           |
| copied   |         |          |           |          |             |             |         |          |           |
| from [2] |         |          |           |          |             |             |         |          |           |
| CpSent   | 0.1876  | 0.4660   | 0.2469    | 0.0674   | 0.1850      | 0.0895      | 0.1230  | 0.2264   | 0.1426    |
| FcFound  | 0.1733  | 0.4389   | 0.2293    | 0.0598   | 0.1672      | 0.0797      | 0.1154  | 0.2168   | 0.1346    |
| CpOnly   | 0.1724  | 0.4034   | 0.2216    | 0.0599   | 0.1537      | 0.0774      | 0.1165  | 0.1996   | 0.1308    |
| MeadLexRank | 0.1629  | 0.4071   | 0.2145    | 0.0569   | 0.1559      | 0.0753      | 0.1092  | 0.2013   | 0.1263    |
| CsSent   | 0.1524  | 0.3871   | 0.2015    | 0.0502   | 0.1420      | 0.0668      | 0.1029  | 0.1934   | 0.1198    |
| Lead     | 0.1432  | 0.3606   | 0.1890    | 0.0487   | 0.1332      | 0.0644      | 0.0985  | 0.1822   | 0.1141    |
| MeadCentroid | 0.1343  | 0.3405   | 0.1777    | 0.0439   | 0.1225      | 0.0584      | 0.0897  | 0.1679   | 0.1043    |
| Random   | 0.1224  | 0.3164   | 0.1624    | 0.0326   | 0.0948      | 0.0436      | 0.0812  | 0.1567   | 0.0952    |
| CsOnly   | 0.1043  | 0.2416   | 0.1319    | 0.0291   | 0.0753      | 0.0371      | 0.0716  | 0.1218   | 0.0790    |

5 Conclusion

We have present our approach for generating catchphrases from legal case documents. Our system is simple in the aspect that it uses only the information inside each document to determine its catchphrases, and extracts catchphrases using preset boundaries (top $t$ anchors, radius $2k$), thereby breaking syntactic constraints. In spite of its simplicity, our system achieves comparable performance compared to methods using corpus-wide and citation information.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR1513, Japan.

References

1. John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
2. Filippo Galgani, Paul Compton, and Achim Hoffmann. Citation based summarisation of legal texts. In Pacific Rim International Conference on Artificial Intelligence, pages 40–52. Springer, 2012.
3. Filippo Galgani, Paul Compton, and Achim Hoffmann. Towards automatic generation of catchphrases for legal case reports. In Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part II, CICLing’12, pages 414–425, Berlin, Heidelberg, 2012. Springer-Verlag.
4. Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 103–112, Denver, Colorado, May–June 2015. Association for Computational Linguistics.
5. Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
6. Yoon Kim. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.
7. Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out, 2004.
8. Yang Liu and Meng Zhang. Neural network methods for natural language processing. Computational Linguistics, 44(1):193–195, 2018.
9. Arpan Mandal, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh. Automatic catchphrase identification from legal court case documents. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM ’17, pages 2187–2190, New York, NY, USA, 2017. ACM.
10. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.
11. L.T. Olsson and Australian Institute of Judicial Administration. Guide to Uniform Production of Judgments. Australian Institute of Judicial Administration, 1999.
12. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.
13. Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 373–382. ACM, 2015.