Large text NLP: Challenges and Solutions

Athens University of Economics and Business

Tsirmpas Dimitris, Gkionis Ioannis

28th November 2022

Contents

1 Introduction

Understanding written text has always been an area of interest for computer research as well as commercial applications. Successfully parsing human text can be crucial for tasks traditionally necessitating lengthy human intervention, such as analyzing sentiments or extracting the underlying meaning and themes in works of literature. During the past few years, research has been pivoting away from more traditional statistical and machine learning methods in favor of emergent neural network and deep learning architectures to tackle this complicated problem.

While these new methods are very promising, they face severe challenges when used in the context of very large documents such as books or scientific articles. In this paper we will examine many of these challenges, how they impact existing solutions and how recent methods in the field of NLP(Natural Language Processing) attempt to circumvent them. We will largely focus on three areas of scientific interest; Text categorization, which attempts to categorize documents into distinct classes, text summarization, which attempts to produce summaries of large documents and sentiment analysis, which attempts to identify the emotional tone behind a body of text.

2 Text Classification

Text classification refers to the process with which we can automatically categorize a document d in one or more classes c based purely on its contents. For example, a program used in a library would use text classification in order to assign labels such as ”Romance” and ”Science Fiction” to a new book.

2.1 Challenges

Text classification algorithms inevitably have to face challenges common to most tasks dealing with natural language processing. Some of these challenges are:

- **The curse of dimensionality**. The curse of dimensionality refers to the tendency for data to become exponentially sparse when projected to high-dimensional spaces compared to low-dimensional ones. Since grouping data is a fundamental procedure in statistics and machine learning, this means we are forced to use an exponentially large set of inputs for our methods to detect and organize the data in meaningful ways. More specifically, in the context of natural language processing, as our dictionary (the set of all unique words in our model) increases in size, the possible combinations of the words within it grow exponentially large, unlike our training samples.

- **Polysemy**. This refers to a word having multiple possible meanings. For example the word ”bank” may mean a financial institution, a building or a synonym for relying upon someone. Our models must learn to differentiate the meaning of words depending on their context.

- **Homonyms**. Homonyms are words that either share the same spelling (homographs), the same pronunciation (homophones) or both. Using our previous example the term ”bank” as a financial institution and the term ”bank” referring to a part of a river are homonyms (the difference between this and Polysemy is that homonyms are not related to each other semantically).

- **Sarcasm**. Sarcasm detection can be seen as a specific case of sentiment analysis, but is important to take account since it often completely inverts the meaning of a phrase.

- **Allegories**. Allegories are especially prevalent in literary works. Their challenge lies in their heavy use of metaphors, which flip the meaning our model has learned.

Literary texts in particular exhibit many additional challenges. First of all, their text is not structured and tends to communicate information in indirect and abstract ways, usually within complicated narratives. In practical terms this means that information crucial to determine the class of a book can be distributed across many chapters[1].

Another, more obvious problem, is the inputs’ size. While CNNs (Convolutional Neural Networks) are considered a state-of-the-art technology, including in the field of Natural Language Processing, they grow proportionally to their inputs. This means that they are unable to process books that feed thousands of paragraphs as input without specialized and highly expensive hardware. Other neural network techniques specialized for NLP such as RNNs (Recurrent Neural Networks) and LSTM (Long-Short Term Memory) networks on the other hand struggle to conserve information over huge spans of text [2].

Furthermore, the language used in literary genres often drifts alongside the genre's own adaptations to social needs. This can lead to deficiencies if the focus of the classification program falls upon a small set of important words or sentences[3].

Finally, during our research we noticed that NLP models typically need to be trained on one language at a time, due to hardware/time constraints or because of their field of use. This might not be an issue for a model specializing in widespread languages such as English, French or Spanish, which contain extensive, supervised datasets of books and literary works, but not for less-spoken languages. This leads to research featuring such languages to favor less data-intensive models, such as decision tree classifiers[3], or artificially augmenting their datasets[4].

2.2 Early Work

Historically two methods were used for the task [5]. In the late 1980s knowledge engineering was used, according to which experts manually inserted a set of rules into the program, which were used to classify the documents. While this meant no training was necessary, the rigidity and human bias of manually inserting rules caused the programs to be unable to scale, generalize and adapt to new documents. Another problem that emerged is that a document could belong to multiple genres that could not be easily separated from each other.

A different approach uses machine learning classification algorithms such as Naive Bayes, SVMs (Support Vector Machines) and ID3 decision trees. Korde and Mahender [6] compile a list of most common machine learning approaches and briefly discuss their upsides, downsides and applications. Notably the K-NN (K-Nearest Neighbors) and Naive Bayes classifiers, while easy to compute and implement, are inefficient when features are highly correlated, Decision Trees can provide excellent insight for decision support tools at the cost of not being memory efficient for large bodies of text and SVMs are highly efficient at multiple categorization but require negative training sets which are harder to acquire. Furthermore they point to LLSF (Linear Least Squares Fit) as being "one of the most effective text classifiers known to date", albeit with a high computational cost.

It's important to note that although new research has largely focused on the emerging applications of neural networks to solve the Text Classification problem, the traditional machine learning methods are still employed [3] [7]. In fact, tree classifiers and logistic regression can be competitive even in large literary texts [8].

2.3 Solutions

Most NLP implementations begin by restricting their model's dictionary by filtering non-statistically significant terms.[8] remove stop words, words that are too frequent/rare in respect to the entire corpus, words that are present in most classes, and choosing only the ones that are present in each book's individual dictionary when training. Worsham and Kalita [1] on the other hand insist on keeping most words in the model's dictionary in order to enable their model to learn patterns that would otherwise be lost, such as tense and plurality.

Another practically necessary step is to limit how much of a document's content will be processed. As mentioned earlier, the computational complexity of neural networks, besides other challenges, prohibit training and testing models with the entire contents of books. Liu et al. [8] sample paragraphs for each document, while [1] use the 5000 first, last or random set of words for each document. Note that all these techniques (apart from random-5000) do not sacrifice the structure of the input's text.

3 Summarization

Automatic Text Summarization (ATS) is the task of generating a short summary for a given document. Importantly, the generated text must be coherent, maintain the most important information, and that information be accurate to the source material. There are two main methods of generating summaries; extractive and abstractive summarization.

Extractive summarization generates its summary using sentences found in the source text. This practically reduces the task of summarization into finding the top-k most important periods and pasting them on the output. This method offers an easy alternative to generating a summary, both algorithmically and computationally. Because of that, extractive summarization is the preferred method for many projects, especially in the context of online articles. However its field of use is limited by the fact that not all document types can be summarized by just a few sentences. For example, there's no way of describing a literary work by using its own sentences.

Abstractive summarization on the other hand generates its own text. This is the method humans use when trying to summarize text, by reading the source, extracting its meaning and then writing down a condensed text that encapsulates as much of that meaning as possible. Modern abstractive techniques mostly use the encoder-decoder pattern to emulate this process. Compared to extractive summarization, this method produces a more meaningful and condensed summary, and is much more adaptable to the kind of document that needs to be summarized. However, because it can no longer rely on the source text's periods as output there's a much larger risk of misrepresenting facts or not being coherent at all.

3.1 Challenges

Extractive models struggle particularly with long-text documents[9]. The longer a document the more topics it typically covers, and the harder it is for a model to produce a summary effectively covering all of them.

One of the most common issues of abstractive models dealing with long-text-documents is sentence repetition. See, Liu and Manning [10] claim that the repetition is caused by the over-reliance of a decoder to its input, causing an endless loop of phrase repetition. Suleiman and Awajan [11] point to issues concerning RNNs with attention mechanisms and also claim that this same issue causes false information to appear in the summaries.

Abstractive models also struggle with recovering words after they have passed through numerous layers of computation See, Liu and Manning [10]. In particular, words that appear infrequently during training are assigned a poor word embedding and as a result are impossible to be retrieved and used in the output.

Furthermore, the ROUGE metric, perhaps the most often used metric in summarization research might not be sufficient for abstractive models. Suleiman and Awajan [11], claim that while this metric is sufficient for extractive summarization, it performs considerable worse in adaptive models since it considers variants of the same word completely separate from each other. They instead advice on using the METEOR metric which can detect variants and synonyms.

Both authors however point out the lack of long-text datasets. They use the CNN/Daily Mail dataset, which as of writing, is the only long-text dataset available. Suleiman and Awajan [11] complain that the dataset includes only highlights of new articles, resulting in many crucial points not being presented in the summary. They also mention the complete lack of datasets for many languages such as Arabic, a recurring problem mentioned above in this paper.

3.2 Early Work

The first studies concerning ATS began as early as 1958. In their paper Luhn [12] used an extractive method to, for the first time, build a summary of technical papers and magazine articles. In 1995 Maybury [13] proposed a new system, SumGen, which can use domain specific knowledge from a database to enhance the summarization process. Of course, these models relied on statistical algorithms such as ”Latent Semantic Analysis” (LSA) and machine learning classifiers such as SVD. It was not until the development of techniques like seq2seq learning and unsupervised language models that allowed neural networks to be applied for abstractive summarization. So far they are the only competitive models that can be applied for this kind of summarization.

3.3 Solutions

[14] are the first to use neural networks, and specifically the encoder-decoder pattern with an attention mechanism to build an extractive summarization model.

Their solution is succeeded by Nallapati, Zhai and Zhou [15] who use ”SummaRuNNer”, a RNN-based sequence classifier with a training mechanism that allows it to train on abstractive summaries. More specifically, during training they take the content, salience, novelty, and position of each sentence into consideration when deciding it should be included in the extractive summary.

Finally, Xiao and Carenini [9] improve on the above solution by incorporating local and global context information, inspired by the hierarchical structure that most human, long documents use. More specifically, three components are used; the sentence encoder, which maps word embedding to a fixed length vector, the document encoder which uses a bi-directional RNN to encode the sentences and a decoder that produces the summary. The decoder chooses between concatenating the sentence vectors produced by the document encoder and uses an attention mechanism to assign weights to them. Their model has been proved to achieve state-of-the-art results in the known CNN/Daily Mail datasets and their efficiency increases with datasets of ever increasing document length.

See, Liu and Manning [10] propose a new solution to the copying problem by using ”pointer-generator networks”. Inspired by the ability of mammals, including humans, to refer to objects they don’t know by pointing at them, pointer-generator networks may choose whether to generate a word, or copy it from the source text.

An important strength of the network is that it can copy out-of-vocabulary words, which makes it possible to generate text with rare words, as well as keep a smaller vocabulary, reducing computational and memory costs.

In a way, this approach combines extraction and abstraction, in order to combine the best of both worlds. Empirically it also appears that the network is faster to train than a traditional seq2seq attention system.

4 Sentiment Analysis / Opinion Mining

Opinion mining (OM) is a field of research that uses Natural Language Processing in order to get information from a text that helps in extracting ideas and opinions and presenting the information in an efficient way[16]. In its earliest forms, it was used mostly in small texts such as reviews and tweets, where the opinion of the author is clearly stated, however in recent years it has started seeing more and more usage in large texts.

A similar field of research that according to some researchers is a subset of OM, and according to others is exactly the same thing as OM[17], is **Sentiment Analysis**(SA). Sentiment Analysis uses NLP approaches to determine the emotions behind a sentence or series of sentences. It has also mostly been used for small texts and reviews[18], however it can and most likely will be widely used in analyzing literature texts, for example to determine characters’ emotional states throughout a book, which can in return help critics categorize these texts.

OM/SA for literature texts is a field that is even nowadays largely unexplored, however it is clear that it can provide useful solutions to problems mentioned in 2.1.

4.1 Challenges

Most of the techniques used in OM/SA use a bag of words known as ”sentiment lexicon”. These are words that are heavily weighed towards positive or negative emotions, for example words such as ”good, bad, great, terrible etc.”[19]. This has been helpful for researchers on a baseline level but presents them with a few important problems such as:

1. Sarcasm: As with text classification, sarcasm and non-literal use of such words creates a problem where the models falsely label a sentence because of these words appearing in it, without understanding their usage.

2. Lack of sentiment lexicon words: In texts that use complicated vocabulary, we often find ourselves lacking these key words and most systems fail to classify the emotions behind these sentences correctly

4.2 Solutions

One of the earliest forms of sarcasm detection was the famous 6-tuple representation. Proposed by Ivanke and Pexman [20] in 2003, it was a very important milestone in sarcasm identification in linguistics. They defined sarcasm consists of 6-tuples where: S= Speaker, H= Listener, C= Content, u= Utterance ,p= Literal proposition, p'= Intended proposition. This can be read as Speaker S generates an utterance u in Context C meaning proposition p but intending that hearer H understands p'.

From as early as 2006[21], researchers have been trying to find more efficient ways to detect sarcasm. Most have used social media sites such as Twitter to train their models[22], paying special attention to the amount of hyperboles used in sarcastic sentences to aid them. In recent years, studies have been able to reach high accuracy in identifying sarcasm, for example a 2017 study on twitter sarcasm detection using Deep Convolutional Neural Networks[19] is able to reach upwards of 90% accuracy.

References

[1] Joseph Worsham and Jugal Kalita. 'Genre Identification and the Compositional Effect of Genre in Literature'. In: *Proceedings of the 27th International Conference on Computational Linguistics* (20th Aug. 2018).

[2] Joseph Worsham. *Towards Literary Genre Identification: Applied Neural Networks for Large Text Classification.* 2014.

[3] Monte Serrat, Mateus Tarcinalli Machado and Evandro E S Ruiz, eds. *A machine learning approach to literary genre classification on Portuguese texts: circumventing NLP's standard varieties.* 2021.

[4] ΙΦΙΓΕΝΕΙΑ ΘΕΟΔΩΡΙΔΟΥ. 'Ανάπτυξη και εφαρμογή μοντέλων γλώσσας σε ελληνικά λογοτεχνικά κείμενα'. ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ, 2020.

[5] Rami Aly. 'Hierarchical writing genre classification with neural networks'. bachelor. University of Hamburg, 18th Oct. 2018.

[6] Vandana Korde and C Namrata Mahender. 'TEXT CLASSIFICATION AND CLASSIFIERS: A SURVEY'. In: *International Journal of Artificial Intelligence & Applications (IJAIA)* (1st Mar. 2012).

[7] Baoxun Xu et al. 'An Improved Random Forest Classifier for Text Categorization'. In: *J. Comput.* 7 (2012), pp. 2913–2920.

[8] Sicong Liu et al. 'DeepGenre: Deep Neural Networks for Genre Classification in Literary Works'. Language Technologies Institute, Carnegie Mellon University.

[9] Wen Xiao and Giuseppe Carenini. 'Extractive Summarization of Long Documents by Combining Global and Local Context'. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3011–3021. DOI: 10.18653/v1/D19-1298. URL: https://aclanthology.org/D19-1298.

[10] Abigail See, Peter J. Liu and Christopher D. Manning. 'Get To The Point: Summarization with Pointer-Generator Networks'. In: *CoRR* abs/1704.04368 (2017). arXiv: 1704.04368. URL: http://arxiv.org/abs/1704.04368.

[11] Dima Suleiman and Arafat Awajan. 'Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges'. In: *Mathematical Problems in Engineering* 2020 (Aug. 2020). DOI: 10.1155/2020/9365340.

[12] H. P. Luhn. 'The Automatic Creation of Literature Abstracts'. In: *IBM Journal of Research and Development* 2.2 (1958), pp. 159–165. DOI: 10.1147/rd.22.0159.

[13] Mark T. Maybury. 'Generating summaries from event data'. In: *Information Processing & Management* 31.5 (1995). Summarizing Text, pp. 735–751. ISSN: 0306-4573. DOI: https://doi.org/10.1016/0306-4573(95)00025-C. URL: https://www.sciencedirect.com/science/article/pii/030645739500025C.

[14] Jianpeng Cheng and Mirella Lapata. 'Neural Summarization by Extracting Sentences and Words'. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 484–494. DOI: 10.18653/v1/P16-1046. URL: https://aclanthology.org/P16-1046.

[15] Ramesh Nallapati, Feifei Zhai and Bowen Zhou. 'SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents'. In: *CoRR* abs/1611.04230 (2016). arXiv: 1611.04230. URL: http://arxiv.org/abs/1611.04230.

[16] Waqar Ahmad and Maryam Edalati. *Urdu Speech and Text Based Sentiment Analyzer*. 2022. DOI: 10.48550/ARXIV.2207.09163. URL: https://arxiv.org/abs/2207.09163.

[17] Rushlene Kaur Bakshi et al. 'Opinion mining and sentiment analysis'. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (2016), pp. 452–455.

[18] Sérgio Barreto et al. *Sentiment analysis in tweets: an assessment study from classical to modern text representation models*. 2021. DOI: 10.48550/ARXIV.2105.14373. URL: https://arxiv.org/abs/2105.14373.

[19] Shelly Sachdeva and Devpriya Soni. 'SCPE DETECTION OF SARCASM IN TEXT DATA USING DEEP CONVOLUTIONAL NEURAL NETWORKS'. In: 2017.

[20] Stacey L. Ivanko and Penny M. Pexman. ‘Context Incongruity and Irony Processing’. In: *Discourse Processes* 35.3 (2003), pp. 241–279. DOI: 10 . 1207 / S15326950DP3503 “ 2. eprint: https : / / doi . org / 10 . 1207 / S15326950DP35032. URL: https://doi.org/10.1207/S15326950DP3503 2.

[21] Joseph Tepperman, David Traum and Shrikanth Narayanan. ‘Yeah right: Sarcasm recognition for spoken dialogue systems’. In: Jan. 2006.

[22] Dmitry Davidov, Oren Tsur and Ari Rappoport. ‘Semi-Supervised Recognition of Sarcasm in Twitter and Amazon’. In: *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 107–116. URL: https:// aclanthology.org/W10-2914.