Abstractive Summarization and Extractive Generalization with NLP. 2022-10033-Standard Purchase Framework Agreement-Livedata Ltd-AI Researchers-P2021-065. Final report.

Livedata Limited

September 22, 2022

Contents

1 Abstract

The research covers two specific Natural Language Processing (NLP) areas: summarisation and generalisation.

The summarisation has been further separated into two distinct sections: extractive and abstractive summarisation (see section A). The former involves locating sentences that are fundamental for understanding the text and copying them into the final output, whereas abstractive summarisation analyses the whole passage and constructs a new output. Notably, extractive summarisation works much faster than its abstractive counterpart, due the simpler algorithm that lies at the base of the code. Moreover, this approach guarantees that the phrase's meaning is not altered. Yet, some important information may be lost, as only a specifically selected number of phrases is extracted. On the other hand, the primary benefit of using abstractive summarisation is that the output is generated based on the meaning of the whole text, significantly decreasing the chance of losing key supplementary information. Thus, abstractive summarisation is more advanced, albeit this comes at the expense of higher complexity - much more computational resources and a well-prepared model are paramount for success. Both approaches have been explored and assessed in the paper and can be tested in the application.

The generalisation task involves removing sensitive or company-specific information from the text. One of the approaches utilised included a combination of an identification-specific part of speech (POS) - such as financial numbers, geographical location and name of companies - (referred to as Named Entities Recognition[1]) and linguistic analysis through dependency parsing to find the specific 'head' words[2]. This approach has been assessed during the research (see section 4.5.2) and deployed in the application.

A wide range of available methodologies and approaches has been tested; the recommendation can be found in the section 4.7.

Furthermore, following the Agile methodology[3], a Class and a UI application have been created and deployed with source code located at GitHub.

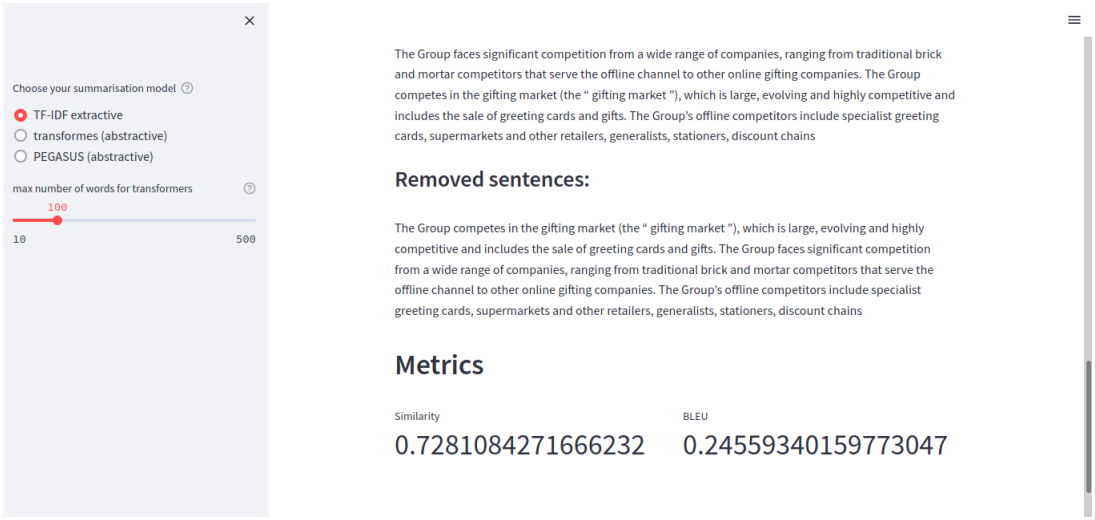

Figure 1: UX application

2 Intro

Currently, there is a wide variety of Natural Language Processing (NLP) tasks; these range from a very basic pattern search to profoundly intelligent chat bots with the functionality to replace human

[1] A named entity is a "real-world object" that's assigned a name – for example, a person, a country, a product or a book title. spaCy can recognize various types of named entities in a document, by asking the model for a prediction. Because models are statistical and strongly depend on the examples they were trained on, this doesn't always work perfectly and might need some tuning later, depending on your use case

[2] spaCy uses the terms head and child to describe the words connected by a single arc in the dependency tree. The term dep is used for the arc label, which describes the type of syntactic relation that connects the child to the head (SpaCy, 2022)

[3] Our highest priority is to satisfy the customer through early and continuous delivery of valuable software (Agile Group, 2001)

operators.

Processing raw text intelligently is difficult: most words are rare, and it's common for words that look completely different to mean almost the same thing. The same words in a different order can mean something completely different. Even splitting text into useful word-like units can be difficult in many languages. While it's possible to solve some problems starting from only the raw characters, it's usually better to use linguistic knowledge to add useful information (SpaCy, 2022).

In particular, there is a specific set of tasks which enables us to process an especially large volume of papers - either in pdf or html format – in order to extract the most important information. Additionally, it is sometimes necessary to remove certain information from the text to follow GDPR or other regulations.

2.1 Project beneficiary (10be5, 2022)

The beneficiary company (the company) was founded in November 2019 with a mission to help companies get listed, and maintain that listing, by automating prospectus drafting.

One of the company's products aims to automate drafting of the risk factors section. In brief, the company uses natural language processing (NLP) to classify individual risk factors from past deals and returns those falling under selected categories to the user.

The company wishes to develop, in collaboration with the others, additional features that would customise the returned risk factors for the deal the user is working on:

- The removal of precedent-specific information. A precedent risk factor typically contains information that is relevant only to the company in the precedent. In order to "generalise" the precedent risk factor, such information must be removed.
- The addition of relevant information. The "generalised" risk factor must be developed to include information relevant to the deal the user is working on. The user may provide such information in bullet points for expansion in the style of the "generalised" risk factor.

The relevant NLP techniques include text summarisation (abstractive and/or extractive), paraphrasing, and generation.

2.2 Example

Suppose the precedent risk factor text is as follows: *"Our business, financial condition and results of operations may be adversely affected by the COVID-19 pandemic.*

The COVID-19 pandemic, and measures to prevent its spread, may have a material adverse impact on our business, financial condition and results of operations. Governmental and non-governmental initiatives to reduce the transmission of COVID-19, such as the imposition of restrictions on work and public gatherings and the promotion of social distancing, have impacted and could continue to impact our operations and financial results. *For example, in the six months ended September 30, 2020, our Trade Show business line ran seven trade shows (versus 21 trade shows in the prior comparable period in 2019) and generated €7.1 million in revenue (versus €31.0 million in the prior comparable period in 2019).* In France for example significant restrictions and social distancing measures remain in place, which continue to adversely affect the overall French and global economies and, in turn, both our French and global operations."

The text *highlighted* in red are issuer-specific and should be removed.

Suppose the user provides the following additional factual information for the deal they are working on:

- Number of exhibitions: 15 in 2019 vs. 2 in 2020
- Number of new leads from in-person sales: around 250 in 2019 vs 20 in 2020

The updated risk factor taking into account this additional factual information could look as follows with the additional relevant information *highlighted*: "Our business, financial condition and results of operations may be adversely affected by the COVID-19 pandemic. The COVID-19 pandemic, and measures to prevent its spread, may have a material adverse impact on our business, financial condition and results of operations. Governmental and non-governmental initiatives to reduce the transmission of COVID-19, such as the imposition of restrictions on work and public gatherings and the promotion of social distancing, have impacted and could continue to impact our operations and financial results.

For example, we were only able to run 2 exhibitions in 2020, compared to 15 in 2019. The number of new leads from in-person sales also decreased dramatically from around 250 in 2019 to 20 in 2020. In France for example significant restrictions and social distancing measures remain in place, which continue to adversely affect the overall French and global economies and, in turn, both our French and global operations."

3 Planning phase

3.1 Problem definition

Thus, the primary aims of this project are:

- Exploration of "abstractive" text summarisation and "extractive" text summarisation options, starting from the ground up and working towards the achievement of custom models - please find the term definition in the Glossary at section A.
- Development of models during the exploration process (it is difficult to predict high accuracy).
- Construction of prelimenary and very basic library/API prototype(s) where appropriate in Python and other programming languages.
- Suitable test metric to assess the performance of the prototype solution (may not be possible for abstractive text summarisation).
- Creation of a brief report with details regarding the methods explored, test results and best suggestions for further improvement.

However, this project does NOT include the following:

- A thorough technical review of all options. Instead, it will focus on allocating more time for the selection of promising options for detailed exploration and subsequent development.
- Data collection[4], since data has already been collected and provided.
- Text insertion problem or "re-specification" of generalised text.

3.2 Key Stages

The project can be divided into the three stages described below. These stages correlate with the Order Form. Currently, the first stage is fully completed, whilst the second and third stages are in progress.

- *Planning and meetings*
  - Meetings with the client and Luke Ellison (from Digital Catapult) to establish the initial scope of the project, gather information, deliver updates and present final reports.
  - Individual planning time and creation of plan[5].
- *Individual Research* (see section 4 below)
  - A body of work to perform the required research. This includes background reading, data analysis, model creation, experimentation and analysis of results.
  - To be conducted in June, July and August 2022.
- *Report creation and work publication*
  - Production of a document outlining the results of the research collaboration and any work required in presenting or publishing these results.
  - A draft will be written prior to 1st August 2022, with the final report completed by mid-September.

[4] The training dataset is available online and sample data is presented in the attachment.
[5] Mostly conducted in the last 2 weeks of May 2022, with some ongoing meetings for updates provision and further consultation when needed

4 Research phase

4.1 Methodology

The project is expected to be divided into the following stages:

- obtaining the full training dataset from the client
- identifying the list of libraries or modules to be used for resolving abstractive summarisation and extractive generalisation tasks
- defining the set of metrics to be used for an assessment of model quality
- training and assessing a set of distinct models, with special consideration given to the most appropriate metrics of performance
- selecting one or more models that are best suited for resolving the necessary tasks
- reaching a conclusion regarding package selection and subsequent model implementation for the NLP task at hand.

4.2 Libraries that suit the summarisation task

Ten of the most popular NLP packages are outlined in the article (kleiber, 2018). At the time of writing, searching for "NLP" on PyPI leads to more than 1,560 results, with similar results for Libraries.io, given that 450 Python packages/libraries are found for the keyword. The principal change which occured in the last few years is the more widespread adoption of neural networks, deep learning, and particularly Transformer-based models, such as BERT (late 2018), GPT-2 (2019), and the relatively recent GPT-3 (2020). Consequently, although 'traditional' approaches to NLP and computational linguistics are still widely utilised as viable solutions, the new models and methods are swiftly moving the entire field forward, enabling us to perform complex tasks via implementing the encoder-decoder architecture.

The following criteria were employed to select the most promising libraries:

- The library is Python-based - Python is the most popular language for all data science problems, including Natural Language Processing
- The library is actively being developed and supported – Therefore, this ensures that the library remains reliable and suitable for professional usage in projects.
- The library is multi-purpose – This enables us to utilise the library for various tasks, meaning that the need to switch between distinct libraries is eliminated.
- The library possesses numerous GitHub stars - this reflects the popularity of a package and guarantees that the library is regularly tested by the GitHub community and likely contains few errors.

Based on this criteria, it is reasonable to conclude that the following ten packages are best suited for Natural Language Processing[6] (kleiber, 2020):

- *Transformers*: Provide easy access to a large number of powerful models (Transformer-based) that can be utilised directly or fine-tuned using PyTorch or TensorFlow.
- *spaCy*: A powerful, all-purpose library with an intuitive API, an excellent ecosystem and sophisticated support for Deep Learning.
- *Gensim*: Excellent for highly-efficient and scalable topic and semantic modelling. Moreover, it provides easy access to Word2Vec and fastText.
- *NLTK*: The principal NLP library that has served as the basis for all other libraries. NLTK is still the de-facto standard for many NLP tasks and continues to be most effective for educational purposes.
- *flair*: Very easy-to-use, whilst also providing powerful features, such as stacking embeddings and custom features such as "Flair embeddings."

[6] The 'GitHub stars criteria' was used for sorting purposes

- *AllenNLP*: Enables the construction of more sophisticated and accurate NLP models through high-level abstractions. Furthermore, it has a very good CLI (Command-Line-Interface) for running pre-trained models.
- *TextBlob*: An ergonomic toolkit that is most effective at solving common, more 'traditional' NLP tasks.
- *Stanza*: Provides a complete and robust NLP pipeline, whilst also serving as an efficient all-purpose library. It is language-agnostic and currently supports 66 languages with pre-trained neural models; this includes access to Stanford's CoreNLP.
- *NLP Architect*: Allows for exploration and experimentation with various state-of-the-art techniques.
- *Spark NLP*: Enables us to run complex NLP pipelines in a distributed environment and contains over 220 pre-trained pipelines and models that can be easily utilised for resolving various tasks.

4.3 Model metrics

There are both standard and specific quality metrics (medium, 2022) that can be used to assess model quality.

4.3.1 Standard metrics

It's critical to use the appropriate metric when assessing machine learning (ML) models. To better understand each metric and the applications they might be used for, we thought it could be beneficial to present a summary of the most common metrics in this paper. Different metrics are proposed to evaluate ML models in different applications. A subset of the metrics presented in this paper may be used to evaluate a models in some scenarios where looking at just one measure won't provide you a complete picture of the issue you are trying to solve.

We group these metrics into different categories based on the ML model/application they are mostly used for, and cover the popular metrics used in the following problems (Shervin Minaee, 2018):

- Classification Metrics (accuracy, precision, recall, F1-score, ROC, AUC etc.)
- Regression Metrics (MSE, MAE)
- Ranking Metrics (MRR, DCG, NDCG)
- Statistical Metrics (Correlation)
- Computer Vision Metrics (PSNR, SSIM, IoU)
- NLP Metrics (Perplexity, BLEU score)
- Deep Learning Related Metrics (Inception score, Frechet Inception distance)

There is no need to mention that there are various other metrics used in some applications (FDR, FOR, hit@k, etc.), which was skipped here.

4.3.2 NLP metrics

Whenever we build NLP models, we need some form of metric to measure the goodness of the model. Bear in mind that the "goodness" of the model could have multiple interpretations, but generally when we speak of it here we are talking of the measure of a model's performance on new instances that weren't a part of the training data (Karan, 2021).

Governing whether the model being used for a specific task is successful or not depends on 2 key factors:

- Whether the evaluation metric we have selected is the correct one for our problem.
- If we are following the correct evaluation process.

Below we will cover some top metrics that we should consider to capture biases.

1. BLEU (stackoverflow, 2022)

   BLEU: Bilingual Evaluation Understudy or BLEU is a precision-based metric used for evaluating the quality of text which has been machine-translated from one natural language to another by computing the -gram overlap between the reference and the hypothesis. In particular, BLEU is the ratio of the number of overlapping -grams to the total number of -grams in the hypothesis. To be precise, the numerator contains the sum of the overlapping -grams across all the hypotheses (i.e., all the test instances) and the denominator contains the sum of the total -grams across all the hypotheses (i.e., all the test instances). Here, each is summed over all the hypotheses, thus, BLEU is called a corpus-level metric, i.e., BLEU gives a score over the entire corpus (as opposed to scoring individual sentences and then taking an average). Notebook with examples can be downloaded from GitHub here

2. NIST

   The name NIST comes from the organization, "US National Institute of Standards and Technology". This metric can be thought of as a variant of BLEU which weighs each matched -gram based on its information gain(Entropy or Gini Index). The information gain for an -gram made up of words 1, .., is computed over the set of references. It is based on the BLEU metric, but with some alterations. Where BLEU simply calculates n-gram precision adding equal weight to each one, NIST also calculates how informative a particular n-gram is. The idea here is to give more credit if a matched -gram is rare and less credit if a matched -gram is common which reduces the chance of gaming the metric by producing trivial -grams.

3. METEOR

   There are two major drawbacks of BLEU:

   - It does not take recall into account.
   - It only allows exact -gram matching.

   To overcome these drawbacks, METEOR (Metric for Evaluation of Translation with Explicit ORdering) came into being which is based on F-measure and uses relaxed matching criteria. In particular, even if a unigram in the hypothesis does not have an exact surface level match with a unigram in the reference but is still equivalent to it (say, is a synonym) then METEOR considers this as a matched unigram.

   More specifically, it first performs exact word mapping, followed by stemmed-word matching, and finally, synonym and paraphrase matching then computes the F-score using this relaxed matching strategy.

   METEOR only considers unigram matches as opposed to -gram matches it seeks to reward longer contiguous matches using a penalty term known as 'fragmentation penalty'. To compute this, 'chunks' of matches are identified in the hypothesis, where contiguous hypothesis unigrams that are mapped to contiguous unigrams in a reference can be grouped together into one chunk. Therefore, longer -gram matches lead to a fewer number of chunks, and the limiting case of one chunk occurs if there is a complete match between the hypothesis and reference. On the other hand, if there are no bigram or longer matches, the number of chunks will be the same as the number of unigrams.

4. ROUGE

   ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is Recall based, unlike BLEU which is Precision based. ROUGE metric includes a set of variants: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. ROUGE-N is similar to BLEU-N in counting the -gram matches between the hypothesis and reference. This is a set of metrics used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation. This metric is able to utilise any Python library (wikipedia, 2022); the python library to calculate such a metric is available here.

5. CIDEr

   CIDEr (Consensus-based Image Description Evaluation) proposed in the context of image captioning where each image is accompanied by multiple reference captions. It is based on the premise that -grams that are relevant to an image would occur frequently in its set of reference captions.

CIDEr weighs each -gram in a sentence based on its frequency in the corpus and in the reference set of the particular instance, using TF-IDF (term-frequency and inverse-document-frequency). However, -grams that appear frequently in the entire dataset (i.e., in the reference captions of different images) are less likely to be informative/relevant and hence they are assigned a lower weight using inverse-document-frequency (IDF) term.

6. SPICE

   SPICE (Semantic Propositional Image Caption Evaluation) is another Image captioning Algorithm that focuses on -gram similarity, here more importance is given to the semantic propositions implied by the text. SPICE uses 'scene-graphs' to represent semantic propositional content. The hypothesis and references are converted into scene graphs and the SPICE score is computed as the F1-score between the scene-graph tuples of the proposed sentence and all reference sentences. For matching the tuples, SPICE also considers synonyms from WordNet similar to METEOR. One issue with SPICE is that it depends heavily on the quality of parsing and it also neglects fluency assuming that the sentences are well-formed. It is thus possible that SPICE would assign a high score to captions that contain only objects, attributes, and relations, but are grammatically incorrect.

   The embedding-based metrics discussed above use static word embeddings, i.e., the embeddings of the words are not dependent on the context in which they are used but here the embedding of a word depends on the context in which it is used.

7. SpaCy similarity

   By default it's cosine similarity, with vectors averaged over the document for missing words.

8. BERT

   BERT is to obtain the word embeddings and shows that using contextual embeddings along with a simple average recall-based metric gives competitive results. The BERT score is the average recall score overall tokens, using a relaxed version of token matching based on BERT embeddings, i.e., by computing the maximum cosine similarity between the embedding of a reference token and any token in the hypothesis.

9. Bert Score

   Bert score or Bidirectional Encoder Representations from Transformers compute cosine similarity of each hypothesis token with each token in the reference sentence using contextualized embeddings. They use a greedy matching approach instead of a time-consuming best-case matching approach and then compute the F1 measure. For more information about Bert, click here. The example of code can be found here

10. MOVERscore

    MOVERscore takes inspiration from WMD metric to formulate an optimal matching metric, which uses contextualized embeddings to compute the Euclidean distances between words or -grams. In contrast to BERTscore which allows one-to-one hard matching of words, MoverScore allows many-to-one matching as it uses soft/partial alignments, similar to how WMD allows partial matching with word2vec embeddings. It has been shown to have competitive correlations with human judgments in 4 NLG tasks: machine translation, image captioning, abstractive summarization, and data-to-text generation.

4.3.3 Conclusion

The customer provided us with a training dataset for both abstractive summarization and extractive generalisation.

We proceeded to conduct a number of tests to gauge the effectiveness of libraries from the list mentioned at section 4.2 above, in order to find two of the most applicable libraries for reaching the goals described above. Flexibility and transformation accuracy were considered to be the most criteria during the testing process.

A specific set of metrics was identified to confirm or disprove the applicability of the model and library for completing the task at hand.

After a comprehensive set of experiments, we concluded that the usage of a cosine-based similarity criteria was most efficient for assessing the quality of the *abstractive summarisation* model.

Similarity is determined by comparing word vectors or "word embeddings", multi-dimensional meaning representations of a word. Word vectors can be generated using an algorithm like word2vec. Computing similarity scores can be helpful in many situations, but it's also important to maintain realistic expectations about what information it can provide. Words can be related to each other in many ways, so a single "similarity" score will always be a mix of different signals, and vectors trained on different data can produce very different results that may not be useful for your purpose. The similarity of Doc objects defaults to the average of the token vectors (SpaCy, 2022).

The *extractive generalisation* result can be explained due to other criteria - such as BLEU or BERTScore – being either more oriented towards assessing translation tasks (assessing the number of the same or similar words when comparing results to expectations). Finally, BLEU has been selected as a main metric for such kind of task.

BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU (Wikipedia, 2022).

Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. Intelligibility or grammatical correctness are not taken into account (Wikipedia, 2022).

BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts. Few human translations will attain a score of 1, since this would indicate that the candidate is identical to one of the reference translations. For this reason, it is not necessary to attain a score of 1. Because there are more opportunities to match, adding additional reference translations will increase the BLEU score (Wikipedia, 2022).

Taking into account that we aim to locate the best match between generated (excluded, in our case) and expected sentences, BLEU was assumed to be the best candidate for *extractive generalisation* metric.

4.4 Modelling

Let us recall the highest priority task for the project: "Exploration of 'abstractive' text summarisation and 'extractive' text summarisation options. Starting from the ground up and working towards custom models." (medium, 2022a)

To resolve the summarisation task two approaches can be used:

- The first approach entails the selection of the most 'important' sentences through a word frequency analysis; the system subsequently selects and copies a selection of sentences from the source without any changes to the sentences. The benefits of using this approach are: (a) the sentences usually make more sense in comparison to a machine-generated alternatives, since they come from the original text and likely do not require additional alignment. (b) the method is sufficiently simple and swift in realising the solution – it enables the usage of very basic NLP packages such as NLTK or even RegExpr to fully resolve the problem. Extractive summarisation based on an assessment of 'sentence importance' is realised into python library pysummarization.

- Another approach is based on Deep Neural Networks (DNNs). The architecture of DNNs is derived from that of the human brain, whereby layers of neurons are intertwined with each other through different 'weights' for each connection (i.e. the process of training the network). The weight coefficients are chosen in accordance to specific mathematical functions, so that each input has a unique, desirable output to match the expected result. DNNs are primarily utilised when it is substantially difficult to adhere to a large number of rules and conditions, since DNNs are extremely flexible and contain numerous parameters. The most important neural net architectures (Aurélien Géron, 2019) are feed-forward neural nets for tabular data, convolutional nets for computer vision, recurrent nets and long short-term memory (LSTM) nets for sequence processing, encoder/decoders and Transformers for Natural Language Processing, as well as auto-encoders and Generative Adversarial Networks (GANs) for generative learning.

4.5 Experimentation

4.5.1 Abstractive summarisation

These approaches allow us to produce a new and completely unique text to summarise the source, and it is hence labelled as abstractive summarisation. To determine the accuracy level, we tested all of the packages described above and came to the conclusion that the best results are presented using the Transformers package (through the default BERT (wikipedia, 2022a) library) and the PEGASUS library. The first package shows more flexibility, since it allows us to select the most effective language module for further utilisation and has a less significant size (approximately 1GB less) in comparison to PEGASUS by default. Thus, it is much swifter in operation and contains more sophisticated performance metrics.

The example of *transformers* implementation is presented below.

```
from transformers import pipeline
import os
## Setting to use the bart-large-cnn model for summarization
summarizer = pipeline("summarization")
import pandas as pd
df = pd.read_csv('../data/Dataset.csv')

arr = []
for i in range(len(df)):
    try:
        txt = df.iloc[i][1]
        sm = summarizer(txt, max_length=100, min_length=5,
        ↪  do_sample=False)[0]['summary_text']
        t1 = nlp(txt)
        t2 = nlp(sm)
        r = t1.similarity(t2)
        print(f'{i} out of {len(df)} returns {r:.3f}')
        arr.append(r)
    except Exception as e:
        print(e)
```

The sentence importance analysis example is shown as the following.

```
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
from collections import Counter
from heapq import nlargest

def sm2(txt):
    doc = nlp(re.sub(r'\n',' ',txt))
    keyword = []
    stopwords = list(STOP_WORDS)
    pos_tag = ['PROPN', 'ADJ', 'NOUN', 'VERB']
    for token in doc:
        if(token.text in stopwords or token.text in punctuation):
            continue
        if(token.pos_ in pos_tag):
            keyword.append(token.text)
    freq_word = Counter(keyword)

    max_freq = Counter(keyword).most_common(1)[0][1]
    for word in freq_word.keys():
        freq_word[word] = (freq_word[word]/max_freq)
    freq_word.most_common(5)

    sent_strength={}
    for sent in doc.sents:
```

```
        for word in sent:
            if word.text in freq_word.keys():
                if sent in sent_strength.keys():
                    sent_strength[sent]+=freq_word[word.text]
                else:
                    sent_strength[sent]=freq_word[word.text]

    summarized_sentences = nlargest(3, sent_strength, key=sent_strength.get)
    # summary
    final_sentences = [ w.text for w in summarized_sentences ]
    summary = ' '.join(final_sentences)
    return(summary)

import re
arr2 = []
for i in range(len(df)):
    try:
        txt = df.iloc[i][1]
        sm = sm2(txt)
        t1 = nlp(txt)
        t2 = nlp(sm)
        r = t1.similarity(t2)
        print(f'{i} out of {len(df)} returns {r:.3f}')
        arr2.append(r)
    except Exception as e:
        print(e)
```

The results of similarity metric comparison for different methods are presented below at picture 2 and 3.

4.5.2 Extractive generalisation

Extractive generalisation demands the removal of any [company] specific information from the source. Our findings have shown that such tasks can be resolved using a Named Entity Recognition (NER) approach, which is usually included in many NLP packages by default. After a set of comprehensive tests, we discovered that the best results were shown when using the SpaCy library, as it contains a number of pre-trained language models which consist of numerous embedded functionalities, including tokenization, lemmatization, Point-Of-Speech (POS) extraction and named-entity extraction. In our API at GitHub - which is deployed at https://livedata.link/nlp/upload - we had created an NER-analyser and subsequently removed all of the sentences that contain any organisational or personal names, dates, currencies, geographical locations and other sensitive data. As the task requested a completion of extractive summarisation, this approach has been determined to be the most reasonable, because it provides the highest similarity metric in comparison to other potential approaches.

The implementation of such an approach using python is shown below.

```
import spacy
nlp = spacy.load("en_core_web_sm")

def excl(txt):
    excl = ""
    doc = nlp(txt)
    for i, s in enumerate(doc.sents):
        for token in s:
            #if token.ent_type_ != '' or token.pos_ != '':
            #print(token.text, token.lemma_, token.ent_type_, token.pos_, token.dep_)
            if token.ent_type_ in ['GPE', 'NORP']\
            or token.pos_ == 'NUM'\
            or (token.dep_ == 'ROOT' and token.lemma_ in
            ↪  ['face','compete','include','benefit','evolve',
```

```
                                                    'affect','rely','develop','
                                                    ↪  accelerate','invest',
                                                   'acquire']):
                #print(token.text, token.lemma_, token.ent_type_)
                excl += (s.text+' ')
                #print(f'Excluded sentence {i}: {s}')
                break
    return excl
```

As shown above, a few methods are combined to remove specific information from the text using token attributes:

- removal of geographical entities (GPE)
- deletion of any numbers from identified point-of-speech (NUM)
- erasing sentences which have 'head' keys included in the list. 'Head' word in a sentence is identified using morphological dependency analysis in SpaCy.

The selection of key *'head'* words was found using the following code:

```
def root(txt):
    doc = nlp(txt)
    for i, s in enumerate(doc.sents):
        for token in s:
            if token.dep_ == 'ROOT':
                print(f'Sentence {i} has {token.lemma_} as a root')
```

4.6 Result Analysis

Despite the fact that the most comprehensive models - such as BERT, GPT(s) or PEGASUS - can provide the required functionality, our aim is to provide the most efficient service for the tasks at hand. Following the comprehensive testing process, it can be confirmed that the following hypotheses are correct:

- *SpaCy* (medium, 2022a) has an advanced linguistic module that is capable of undertaking all of the necessary linguistic transformations and analysis. Additionally, it contains a list of modules to be used for heavily individual tasks, such as semantic analysis and entity extraction. This includes the ability to connect any third-party Neural Networks framework, such as TensorFlow or PyTorch. Notably, even though there is extensive functionality provision, SpaCy is relatively more overwhelming in comparison to other libraries.
- *Transformers* is a framework that is utilised to adapt any third-party libraries, including TensorFlow and BERT. Thus, Transformers allows us to efficiently employ all of the privileges of transforming learning, although it is important to note that sometimes this makes the resulting library overweight or too complex for commercial service use.

4.6.1 Abstractive summarisation

Below, you can see the result of BERT-based *transformers* and *SpaCy-based* absractive summarisation models.

The similarity score for abstractive summarisation averages at approximately 0.8, meaning that most of the system-generated phrases have similar average word-based vectors to ones expected to be seen by the client in the training dataset. The similarity score (vertical axis) has a range from zero to one, where one means that the whole text matches and vice versa. The horizontal axis reflects the set of tested texts from the training dataset.

The averaged similarity score for abstractive summarisation is approximately 1.0, which means that almost all of the system-generated phrases have the same average word-based vector as the output expected by the client. As mentioned above, the similarity score (vertical axis) has a range from zero to one, where one means that the whole text matches and vice versa. The horizontal axis reflects the set of tested texts from the training dataset.

Remarkably, it became clear that the extractive summarisation model showed a better similarity metric than the more advanced BERT model.

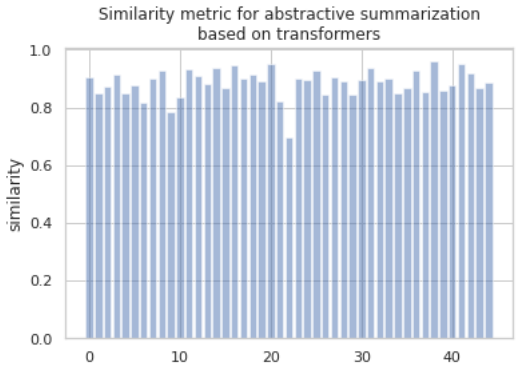

Figure 2: Abstractive (BERT) summarisation similarity metrics

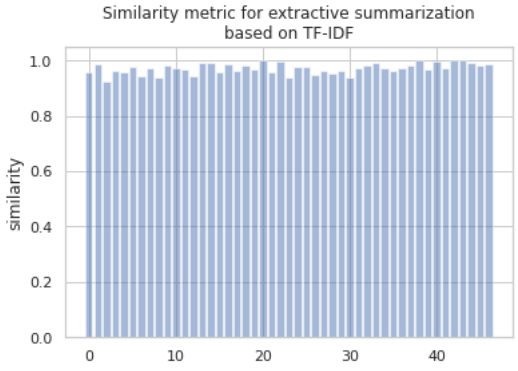

Figure 3: Exstractive summarisation similarity metrics

4.6.2 Extractive generalisation

To resolve the extractive generalisation task, we assumed that *point-of-speech* (POS) and entity-type recognition should be used to optimise performance, whereby all the sentences containing either geographical position, name, or numbers are removed. Notably, the deletion of any organisation names would also be significant in enhancing performance.

To assess the quality of the model the following approach has been utilised:

- the source dataset has been converted into a new one by creating a new column which contains the sentences *to be removed* from the source text.
- the SpaCy-based POS and entity type recognition were used for selecting sentences which have any *specific* information, such as location, name of business, etc.
- the *corpus-based BLEU* metric[7] was used to compare sentences that have been removed with the sentences which are *expected* to be removed.

As a result of the comparison, the following chart was constructed.

As mentioned in section 4.3.3, BLEU works by assessing the degree of word-to-word equivalence and compatibility. That is a task for *extractive generalisation*, as it is expected that all of the sensitive-information-related sentence will be removed.

As illustrated in Fig.4 above, the average BLEU score (vertical axis) for the training dataset was around 0.6, which is an acceptable accuracy for this task. However, the principal issue is that we have a relatively limited number of expected sentences (horizontal axis), thus constraining the accuracy of the assessment.

[7] Corpus score calculation compares 1 candidate document with multiple sentence and 1+ reference documents also with multiple sentences. It is differ from averaging BLEU scores of each sentence, it calculates the score by *summing the numerators and denominators for each hypothesis-reference(s) pairs before the division*

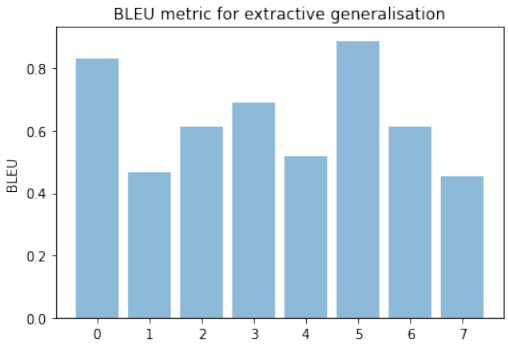

Figure 4: BLEU score for extractive generalisation test

4.7 Conclusion

Following the successful completion of our experiments, we are able to conclude that:

- the most applicable library to be used for *abstractive summarisation* is the 'summarisation' module, based on a BERT library, such as *transformers*.

However, the similarity metric which was used for the training dataset demonstrated that simple 'TF-IDF'-based summarisation works faster than BERT-based abstractive summarisation, displaying higher similarity. The result of the processing is presented above.

- the PEGASUS module has also provided a good result (as shown below), but it requires more resources to ensure effective operation. Hence, it occupies the second spot of our rating.
- at the same time, to complete an *extractive generalisation* task, the SpaCy module has proved to be one of the most effective, as confirmed by the image 4 above.

Additionally, given that we closely follow the Agile methodology, we have developed a Class, simple API and UI application – which is available on GitHub - to assess the training dataset and experiment with any other texts. This is profoundly important because this adds additional flexibility to our model, allowing for further usage in other applications without significant adaptations.

5 Bibliography

10be5 (2022). *NLP project proposal.*
Agile Group (2001). *Agile Manifesto.*
Aurélien Géron (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.*
Karan (2021). *Top Evaluation Metrics For Your NLP Model.*
kleiber (2018). *https://kleiber.me/blog/2018/02/25/top-10-python-nlp-libraries-2018/.*
kleiber (2020). *https://kleiber.me/blog/2020/08/26/top-10-python-nlp-libraries-2020/.*
medium (2022). *https://medium.com/@mikeusru/common-metrics-for-evaluating-natural-language-processing-nlp-models-e84190063b5f.*
medium (2022a). *https://medium.com/swlh/abstractive-text-summarization-with-nlp-ec3924c0b1d5.*
medium (2022a). *https://medium.com/nlplanet/two-minutes-nlp-spacy-cheat-sheet-21471dac7837.*
paperswithcode (2022). *https://paperswithcode.com/task/abstractive-text-summarization.*
Shervin Minaee (2018). *20 Popular Machine Learning Metrics. Part 1: Classification Regression Evaluation Metrics.*
SpaCy (2022). *Linguistic Features.*
stackoverflow (2022). *https://stackoverflow.com/questions/32395880/calculate-bleu-score-in-python.*
wikipedia (2022). *https://en.wikipedia.org/wiki/ROUGE*$_{(metric)}$.
Wikipedia (2022). BLEU.
wikipedia (2022a). https://en.wikipedia.org/wiki/BERT$_{(language_model)}$.

A Glossary

- *Abstractive summarization*: Abstractive Text Summarization is the process of generating a short and conci@ml2ummary that captures the salient ideas of the source text (paperswithcode, 2022). The generated summaries may potentially contain new phrases and sentences that may not appear in the source text.
- *Extractive generalization*: In our context, extractive generalization reflects the removal of any entity-related specific information, such as an organization's name, dates, currency value, etc. The extraction process may either copy non-specific sentences or create a summary based on the key information present.
- *Abstractive*[8]: Rephrasing a block of text.
- *Extractive*: Extracting data from a block of text only

B Source code

B.1 Summ Class

```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
from collections import Counter
import re
from heapq import nlargest
### transformers for abstractive summarisation
from transformers import pipeline
### PEGASUS abstractive summarisation
from transformers import PegasusForConditionalGeneration, PegasusTokenizer
import torch
### BLEU
import nltk
import nltk.translate.bleu_score as bleu

class Summ:
    """Class to manipulate text performing extractive and abstractive summarisation,
        extractive generalisation and providing with metrics
    """

    def __init__(self, text, exp_sum=None, exp_gen=None):
        """Initialise the object for further anipulation.

        Args:
            text (str): the source text to be analysed as a set of strings.
            exp_sum (str): the expected summarisation from the source text, optional.
            exp_gen (str): the expected set of sentences to be removed during
↪   extractive generalisation, optional.
        """

        self.text = text
        self.exp_summ = exp_sum
        self.exp_gen = exp_gen
        self.nlp = spacy.load("en_core_web_sm")


    def ext_summ(self):
```

[8] Extractive summary is choosing specific sentences from the text to compile a summary, while abstractive summary means generating a summary in the computer's own words.

```
        """ Perform extractive summarisation based on TF-IDF

        Returns:
            set of strings (long string) selected as a summary. Sentences are copied
↪   from the source text based on their 'importance' score
        """
        doc = self.nlp(re.sub(r'\n',' ',self.text))
        keyword = []
        stopwords = list(STOP_WORDS)
        pos_tag = ['PROPN', 'ADJ', 'NOUN', 'VERB']
        for token in doc:
            if(token.text in stopwords or token.text in punctuation):
                continue
            if(token.pos_ in pos_tag):
                keyword.append(token.text)
        freq_word = Counter(keyword)

        max_freq = Counter(keyword).most_common(1)[0][1]
        for word in freq_word.keys():
            freq_word[word] = (freq_word[word]/max_freq)
        freq_word.most_common(5)

        sent_strength={}
        for sent in doc.sents:
            for word in sent:
                if word.text in freq_word.keys():
                    if sent in sent_strength.keys():
                        sent_strength[sent]+=freq_word[word.text]
                    else:
                        sent_strength[sent]=freq_word[word.text]

        summarized_sentences = nlargest(3, sent_strength, key=sent_strength.get)
        # summary
        final_sentences = [ w.text for w in summarized_sentences ]
        summary = ' '.join(final_sentences)
        return summary


    def abs_summ_transf(self, max_length=100):
        """ Perform abstractive summarisation based on transformers

        Args:
            max_lengh (int): maximal length of returned summary, in words, optional.

        Returns:
            set of strings (long string)
        """
        summarizer = pipeline("summarization")
        return summarizer(self.text, max_length=max_length, min_length=5,
        ↪   do_sample=False)[0]['summary_text']


    def abs_summ_pegasus(self):
        """ Perform abstractive summarisation based on Google's PEGASUS algorithm

        Returns:
            set of strings (long string)
        """
```

```
        model_name = 'google/pegasus-xsum'
        torch_device = 'cuda' if torch.cuda.is_available() else 'cpu'
        tokenizer = PegasusTokenizer.from_pretrained(model_name)
        model = PegasusForConditionalGeneration.from_pretrained(model_name).to(torch_
        ↪  device)

        batch = tokenizer.prepare_seq2seq_batch([self.text], truncation=True,
        ↪  padding='longest',return_tensors='pt')
        translated = model.generate(**batch)
        return tokenizer.batch_decode(translated, skip_special_tokens=True)[0]


    def similarity(self, sm=None):
        """ Calculate SpaCy based similarity metric

        Args:
            sm (str): string to be compared with

        Returns:
            float: SpaCy similarity score
        """

        if self.exp_summ is not None and sm is not None:
            t1 = self.nlp(self.exp_summ)
            t2 = self.nlp(sm)
            return t1.similarity(t2)
        else:
            return 0


    def generalisation(self):
        """ Return sentences to be removed as they have any sensitive information.

        The approach to be used is a combination of identification specific part of
↪   speech (POS) such as financial numbers, geographical location, name of companies
↪   etc. (called  Named Entities Recognition) and linguistic analysis (dependency
↪   parsing) to find the specific 'head' words.

        Returns:
            str: set of string to be removed from the source text

        """
        excl = ""
        doc = self.nlp(self.text)
        for i, s in enumerate(doc.sents):
            for token in s:
                #if token.ent_type_ != '' or token.pos_ != '':
                #print(token.text, token.lemma_, token.ent_type_, token.pos_,
                ↪  token.dep_)
                if token.ent_type_ in ['GPE', 'NORP']\
                or token.pos_ == 'NUM'\
                or (token.dep_ == 'ROOT' and token.lemma_ in
                ↪  ['face','compete','include','benefit','evolve',
                                                        'affect','rely','develo
                                                        ↪  p','accelerate','in
                                                        ↪  vest',
                                                    'acquire']):
```

```
                    #print(token.text, token.lemma_, token.ent_type_)
                    excl += (s.text+' ')
                    #print(f'Excluded sentence {i}: {s}')
                    break
        return excl


    def root(self):
        """Print root words for each sentence
        """

        doc = self.nlp(self.text)
        for i, s in enumerate(doc.sents):
            for token in s:
                if token.dep_ == 'ROOT':
                    print(f'Sentence {i} has {token.lemma_} as a root')


    def bleu_score(self, ex=None):
        """ Returns BLEU score. Corpus score calculation Compares 1 candidate
        ↪  document with multiple
        sentence and 1+ reference documents also with multiple sentences. Different
↪   than averaging BLEU scores of each sentence, it calculates the score by "summing
↪   the numerators and denominators
        for each hypothesis-reference(s) pairs before the division"

        Returns:
           float: BLEU score from zero to 1
        """

        if self.exp_gen is not None and ex is not None:
            return bleu.corpus_bleu([[ex]], [self.exp_gen])
        else:
            return 0
```

B.2 Application (UI)

```
import streamlit as st
import os
import json

from summ import Summ

st.set_page_config(
    page_title="Extractor",
    page_icon="",
)

st.title("Extractor")
st.header("")

with st.expander("About this app", expanded=True):

    st.write(
        """
        - Currently, there is a wide variety of  Natural Language Processing (NLP)
↪   tasks; these range from a very basic pattern search to profoundly intelligent
↪   chat bots with the functionality to replace human operators.
```

```
        - In particular, there is a specific set of tasks which enables us to process
↪   an especially  large volume of papers - either in pdf or html format - in order
↪   to extract the most important information. Additionally,  it is sometimes
↪   necessary to remove certain information from the text to follow GDPR or other
↪   regulations.
        """
    )

    st.markdown("")

st.markdown("")
st.markdown("## ** Paste document **")

with st.sidebar:
    ModelType = st.radio(
        "Choose your summarisation model",
        ["TF-IDF extractive", "transformes (abstractive)", "PEGASUS (abstractive)"],
        help="At present, you can choose between 3 models to play with your text.
        ↪  More to come!",
        )

    max_length = st.slider(
        "max number of words for transformers",
        min_value=10,
        max_value=500,
        value=100,
        help="It works for transformers abstractive summarisation only",
        )

with st.form(key='form'):
    text = st.text_area(
    "Paste your text below (max 500 words)",
    height=300,
        )

#    MAX_WORDS = 500
#    import re
#    res = len(re.findall(r"\w+", text))
#    if res > MAX_WORDS:
#        st.warning(
#            " Your text contains "
#            + str(res)
#            + " words."
#            + " Only the first 500 words will be reviewed. Stay tuned as increased
↪   allowance is coming"
#        )

#    text = text[:MAX_WORDS]

    exp_summ = st.text_area(
            "Paste your expected summarisation text below (optional)",
            height=100,
        )

    exp_gen = st.text_area(
            "Paste your text to be removed from the source (optional)",
            height=100,
        )
```

```
    submit_button = st.form_submit_button(label="Extract")


if not submit_button:
    st.stop()

if exp_summ == '':
    exp_summ = None
if exp_gen == '':
    exp_gen = None

t = Summ(text, exp_summ, exp_gen)

if ModelType == "TF-IDF extractive":
    summ = t.ext_summ()

elif ModelType == "transformes (abstractive)":
    summ = t.abs_summ_transf()

else:
    summ = t.abs_summ_pegasus()

# extraction
ext = t.generalisation()

# metrics
sim = t.similarity(summ)
bl = t.bleu_score(ext)

print(f'metrics: similarity {sim}, bleu {bl}')

st.markdown("## Check results ")

st.subheader('Source text:')
st.write(text)
st.subheader('Summarisation:')
st.write(summ)
st.subheader('Removed sentences:')
st.write(ext)

st.header("Metrics")
col1, col2 = st.columns(2)

col1.metric('Similarity', sim)
col2.metric('BLEU', bl)
```

C Training dataset examples

```
"Original text", "Abstractive summarization (extremely short version - will require
↪  longer version in practice)", "Extractive generalization (highlighting
↪  information that is company-specific)"
```

```
"The Group competes in the gifting market (the “ gifting market ”), which is large,
↪  evolving and highly competitive and includes the sale of greeting cards and
↪  gifts. The Group faces significant competition from a wide range of companies,
↪  ranging from traditional brick and mortar competitors that serve the offline
↪  channel to other online gifting companies. The Group’s offline competitors
↪  include specialist greeting cards, supermarkets and other retailers, generalists,
↪  stationers, discount chains and florists. The Group also competes with online
↪  greeting card companies; online flower specialists; and online gift specialists.
↪  Some of the Group’s competitors, particularly supermarkets, general merchandise
↪  discounters and stationery retailers, may have larger and broader customer bases,
↪  wider distribution channels, substantially greater financial, technical or
↪  marketing resources, stronger brand or name recognition or a lower cost base than
↪  the Group. Some of the Group’s competitors may have greater research and
↪  development resources and be able to adapt to changes in customer requirements,
↪  customer preferences or attitudes toward design content and gifting products
↪  faster, launch innovative products more quickly, more readily take advantage of
↪  acquisitions and other opportunities, or have more established relationships with
↪  third-party suppliers, which could result in the Group not being able to compete
↪  as effectively and lose its market position. The Group’s competitors may also
↪  aggressively discount their products in order to gain market share, which could
↪  result in pricing pressures, reduced profit margins, lost market share, or a
↪  failure to grow market share for the Group. Within each of the UK and the
↪  Netherlands, the Group has benefited from its strong positions in the gifting
↪  market, but competition could intensify as traditional retailers expand their
↪  digital, online and app-based sales capabilities and potential new competitors
↪  enter the gifting market. The Group could face an increase in online competition
↪  as a result of competitors prioritising investments in new or improved online
↪  platforms to deal with disruptions faced as a result of the novel strain of
↪  coronavirus causing Covid-19 disease (“ Covid-19 ”) and the expected continued
↪  shift to online purchasing, which may make it more difficult for the Group to
↪  maintain its market share. The Group competes, and could increasingly compete in
↪  the future, with alternatives or substitutes to the Group’s products, whether
↪  that is the increasing use of electronic gift cards as substitutes for physical
↪  gifts, social media or other companies that host and enable the posting of
↪  greetings, images, electronic or other gifts, e-cards or other innovations and
↪  developments. In addition, the Group competes and may increasingly compete in the
↪  future with alternative business models that serve the gifting markets, including
↪  greeting card subscription services, flower subscription services, or apps or
↪  websites that provide free products (only charging for any up-sells of
↪  attachments to the free base product) that compete or may serve as a substitute
↪  to the Group’s gifting products. If such alternative communications media or
↪  substitutions for the Group’s products appeal more to the Group’s existing
↪  customers or potential customers and the Group fails to innovate its product
↪  offering in a manner that continues to be attractive to its customers or enables
↪  the Group to maintain its existing margins, the Group may be unable to compete
↪  effectively. If the Group is not able to compete effectively against its current
↪  or potential competitors, this could have a material adverse effect on the
↪  Group’s business, results of operations, financial condition or prospects.",
"The Group faces significant competition for its gifting products, including greeting
↪  cards and gifts, and if the Group does not compete effectively, its business,
↪  results of operation, financial condition or prospects could be materially
↪  adversely affected.",
```

```
"The Group faces significant competition from a wide range of companies, ranging from
↪  traditional brick and mortar competitors that serve the offline channel to other
↪  online gifting companies. The Group's offline competitors include specialist
↪  greeting cards, supermarkets and other retailers, generalists, stationers,
↪  discount chains and florists. The Group also competes with online greeting card
↪  companies; online flower specialists; and online gift specialists. Within each of
↪  the UK and the Netherlands, the Group has benefited from its strong positions in
↪  the gifting market, but competition could intensify as traditional retailers
↪  expand their digital, online and app-based sales capabilities and potential new
↪  competitors enter the gifting market. The Group competes, and could increasingly
↪  compete in the future, with alternatives or substitutes to the Group's products,
↪  whether that is the increasing use of electronic gift cards as substitutes for
↪  physical gifts, social media or other companies that host and enable the posting
↪  of greetings, images, electronic or other gifts, e-cards or other innovations and
↪  developments. In addition, the Group competes and may increasingly compete in the
↪  future with alternative business models that serve the gifting markets, including
↪  greeting card subscription services, flower subscription services, or apps or
↪  websites that provide free products (only charging for any up-sells of
↪  attachments to the free base product) that compete or may serve as a substitute
↪  to the Group's gifting products. If such alternative communications media or
↪  substitutions for the Group's products appeal more to the Group's existing
↪  customers or potential customers and the Group fails to innovate its product
↪  offering in a manner that continues to be attractive to its customers or enables
↪  the Group to maintain its existing margins, the Group may be unable to compete
↪  effectively."
```