Generation of Video Descriptions with an NLG Approach

Jiahao Zhang
DAUIN
Torino, Italy
s301185@studenti.polito.it

Qu Tianming
DAUIN
Torino, Italy
s302398@studenti.polito.it

Flavio Spuri
DAUIN
Torino, Italy
s303657@studenti.polito.it

Luca Agnese
DAUIN
Torino, Italy
l.agnese@studenti.polito.it

Matteo Donadio
DAUIN
Torino, Italy
matteo.donadio@studenti.polito.it

ABSTRACT

By reducing the boredom and confusion often associated with online product searches, our system is designed to enhance the customer experience and help them find the best products with ease. In particular, an automated system generates both textual and video reviews of devices based on their features, leveraging a transformer-based model for text generation. The results of our experiments show that the system is capable of generating reviews of a quality comparable to the ones that can be found in e-commerce platforms, making it a valuable tool for customers looking to make informed decisions.

1 INTRODUCTION

Natural Language Generation (NLG) is a sub-field of Natural Language Processing (NLP), that represents a leading area in AI research and development. NLG is concerned with the automatic creation of written or spoken language and is used in various applications such as content generation, chat-bots, and summarization. The widespread adoption of NLP systems, including those utilizing NLG, has laid the foundation for the creation of innovative applications for natural language understanding.

However, implementing sophisticated models for practical use remains a challenge, particularly in business contexts in which the final output quality has a direct impact on the economic performance of the service. One such area where the application of NLP techniques can have a significant impact is in the generation of product reviews. Nowadays, with the rapid development of technology, tech devices are updating at an unprecedented rate, making it increasingly difficult for consumers to keep up and make informed decisions. This not only causes frustration for consumers but also results in significant costs for e-commerce and related platforms, which must spend a significant amount of time and resources to produce detailed reviews for each product.

In this project, carried out in collaboration with Flowygo, our aim is to develop a system that automatically generates reviews for customers based on a Natural Language Process algorithm, aligning with the **12th SDG**: Sustainable Consumption and Production. To achieve this goal, we have utilized three initial datasets: Croma Electronic products dataset, which is a site dedicated to the online shopping of tech products, and two datasets regarding some tech products of the well-known e-commerce site Amazon: Amazon Phone Dataset, which contains information about phones, and Amazon Product info sample, concerning technological devices. In addition, we have also used a small dataset of manually tagged reviews. The pre-processing step consists in cleaning data, merging the datasets, and splitting the longest descriptions of products. This merged dataset will serve as the training set for our textual generation model.

The methodology involves the use of a Named Entity Recognizer (NER) model to identify the name and attributes of a product, that we then use to generate its review. For textual review generation, we perform a fine-tuning of the GPT-2 model, followed by a post-processing step to improve the quality of the final outcome. At the end of this stage, the complete model for generating product descriptions from structured information is created, which can be integrated into the desired application. The last step is generating a video presentation of the review from the post-processed review generated. This approach allows us to quickly and automatically create reviews for tech devices in an easily digestible format, to be used both by companies and consumers. In the following sections, we will detail the methodology, results, and implications of the project, providing a valuable resource for researchers, industry professionals, and policymakers interested in the application of NLP techniques in real-world scenarios.

2 RELATED WORK

In recent years, Natural Language Generation (NLG) has gained significant attention from both companies and researchers. The ability to automatically generate human-like text has wide-ranging applications, from automating customer service interactions to creating product descriptions for e-commerce platforms.

Our work is a continuation of Andrea Avignone's Master Thesis [1], which deals with the same topic of generating product descriptions from structured data. We have carried on Avignone's work by expanding on the text generation and implementing the video generation.

Additionally, many other researchers and companies have been exploring the use of NLG for similar purposes. Our work has been made possible thanks to the work done by companies that are pioneers of AI transition, such as Google or OpenAI, which have developed foundation models such as GPT-2, which we have used in our project

Overall, our work can be seen as a contribution to the growing body of literature on the use of NLG for e-commerce and other business applications. We aim to demonstrate the potential of these

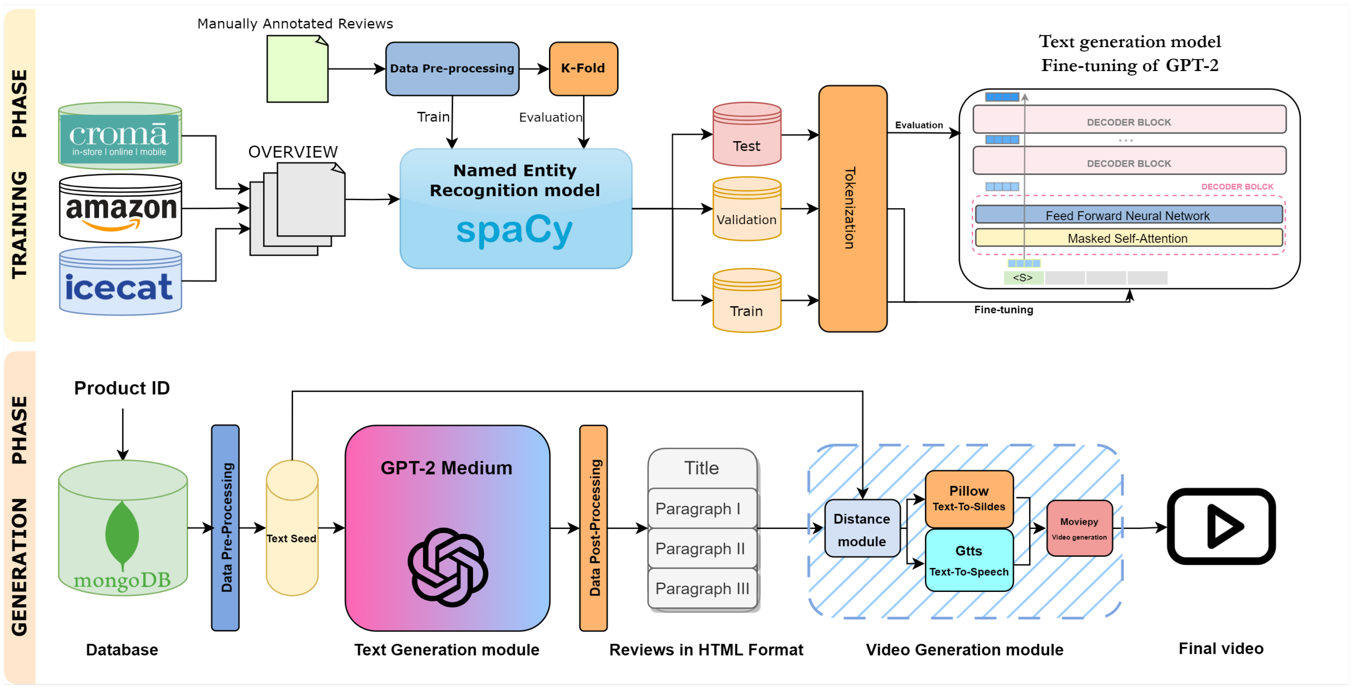


Figure 1: Video Generation Pipeline Diagram

techniques for automating the process of generating product descriptions, and in the process, promoting sustainable consumption and production patterns.

3 METHODOLOGY

In order to implement the Video generation model, we need to design a complete pipeline from the product ID to the final video.

As fig 1 shows, we can split our functional model into two phases:

- **Training Phase**: in this phase, we will prepare a proper dataset for fine-tuning and evaluating our textual generation model, which contains the original dataset preprocessing, the attribute tagging by **Named Entities Recognition** model, and fine-tuning & evaluating operations for Text Generation Model (i.e. the **GPT-2 Model**).
- **Generation Phase**: in this phase, we constructed a pipeline for video generation, which contains the data retrieval from **MongoDB**, the **Text Generation Module**, and the **Video Generation Module**.

In the following section, we will discuss in detail the function and mechanism of each module.

3.1 Customized NER

Being the core purpose of the project, the performance of the text generation model becomes extremely important, and how to prepare a suitable dataset to fine-tune the model becomes a crucial challenge.

Our original dataset is composed of electronic products' information collected from major e-commerce platforms, which is not immediately suitable for fine-tuning the GPT-2 model. To solve this problem, we followed Avignone's [1] line of thinking, i.e. implementing a **Customized Named Entity Recognition** (NER) model to help us extract the product attributes from the original reviews which, combined with some special tokens, are used to build a proper dataset.

The Named Entity Recognition task can be regarded as a token-level classification task, i.e. the algorithm needs to classify every single word (even the punctuation marks) and combine them to extract key information from the text.

So far the majority of the NER models are used for recognizing the person's name or organization, which is useless for our objective. In our case, we need the NER model to extract the **name** and **attributes** of a product from its review. We trained a custom NER model based on **spaCy**, as Fig 1 shows. The spaCy NER system contains a word embedding strategy using sub-word features and "Bloom" embed, and a deep convolution neural network with residual connections. Moreover, the spaCy system builds a complete pipeline from the text to the named entities, which is easy to train and offers sufficiently reliable performance.

To train the NER model, we use manually annotated product reviews (we will introduce more details in Section IV). Since the spaCy is a multiple-task NLP pipeline, we disabled other functions and only kept the NER. Then we use SGD as the optimizer and compute the loss from gradient descent to update the model, setting a dropout rate [2] to avoid over-fitting.

3.2 Text Generation Module

Nowadays, using the **Transformer** architecture to perform NLP tasks has become the standard, and for specific NLG tasks the current models' architectures can be roughly divided into two types:

BERT and GPT. The BERT is based on a Encoder only architecture; on the other hand, the GPT only uses the Transformer Decoder blocks and removes the Encoder-Decoder self-attention layer [3], as Fig 1 shows. This kind of structure makes the GPT model achieve the **Masked Self-attention** and **Auto-Regression** mechanisms, which allow it to generate text more naturally and fluently. This architecture is also more computationally efficient, which allows the GPT to be trained on a much larger dataset than other models. Therefore, we selected the **GPT-2** model as our text generation model. Now, we will discuss in detail how these mechanisms work.

- **Masked Self-attention**: GPT uses sentence sequences to predict the next word, using Masked Self-Attention to cover the context of words coming after it, in order to prevent information leakage. For example, given a sentence containing 4 words, [A, B, C, D], GPT needs to use A to predict B, then uses [A, B] to predict C, and finally [A, B, C] to predict D. While using A to predict B, it will mask information coming from [B, C, D]. This mechanism makes the GPT model more similar to the traditional language model. On the other hand, BERT architecture can use information from all the words in the sentence when predicting the next word, which makes the predicted word affected by the context after it. This is one of the main reasons why BERT has worse performances in the natural language generation task.
- **Auto-Regression**: this mechanism allows the GPT-2 model to add the newly generated token to the sequence of inputs. This new sequence becomes the input to the model in its next step, which makes the generated text more fluent and more "human-like", as it is similar to the way humans write.

During the process of fine-tuning, for the next predicted token, the probability $P(x^i|x^1,...,x^{i-1})$ is computed by applying the **Softmax** function and then the following function is maximized:

$$L_2(C) = \sum_{x,y} \log P(x^i|x^1,...,x^{i-1}) \tag{1}$$

In order to improve the training speed and the generalization ability of the model, the GPT-2 uses Multi-Task Learning and also considers the pre-trained loss function $L_1(C)$ when fine-tuning the model:

$$L_3(C) = L_2(C) + \lambda \times L_1(C) \tag{2}$$

After the fine-tuning, at the text generation stage, we have the **Temperature** parameter T that can directly affect the generated text, which is a parameter of the soft-max function, as the formula shows:

$$P_i = \frac{e^{y_i/T}}{\sum_{k=1}^{n} e^{y_k/T}} \tag{3}$$

In this case, when the temperature is high, the probability of all candidate words will tend to be similar. On the other hand, with a low temperature, the differences in the probabilities for each candidate word will become larger. Different options for temperature parameters will affect the style of the generated texts. In section IV, more detail about the application of different temperature values in different scenarios will be presented.

As a continuation of previous work, we found that they had to truncate too long text because of the device limitation[1], this kind of operation can easily make the training text be misinterpreted and even become meaningless. In this project, we overcame this problem with adequate computing resources. In fact, we choose a larger model than the previously used one: **GPT-2 Medium**, which contains 345M parameters (vs the 117M of the small version), and the result shows the performance has a non-negligible improvement.

3.3 Video Generation

Once a textual review of a given product is generated, some last steps need to be done to generate the corresponding video review. Hence, we will now analyze the pipeline contained in the Video Generation module. In particular, we will further split this module into two sections: a *distance module* and a *video and audio generation module.*

3.3.1 Distance module. Notice that the video consists in a slideshow in which several pieces of information about the considered product are given: in it, we show the image of the product and the logo of the company, as well as features that have eventually been used in the generation of the textual review. Along with this, a text-to-speech will read the textual review, and the audio and video will be synchronized so that each time a sentence containing a certain attribute is read, the attribute will appear on the screen.

Given the format of the video, the main problem was to find a way to synchronize the various information. The purpose of the distance module is to handle this problem before starting the generation of the audio and video.

The input of this sub-module consists of the following:

- the generated textual review
- the preprocessed data that was used as seed for the review generation, i.e. a JSON file containing the list of features that will possibly be used in the review

and the output will be a dictionary containing:

- as keys, the paragraph names
- as values, a nested dictionary that contains information at a sentence-level granularity for the considered paragraph, in particular:
  - as keys, the feature(s) that are found in the considered sentence
  - as values, the sentences

We will now see the various steps needed to obtain the output. Firstly, the input must be converted to a suitable format. An HTML parser is used to break down the textual review in a first dictionary containing the name of the paragraphs as keys and the text of the paragraphs as values, thus moving to a paragraph-level granularity. Then, each paragraph is broken down into sentences, and in each sentence, the eventually mentioned attributes need to be identified. I.e., we need to compare the current sentence to the list of features coming from the JSON file. To do so, we took into consideration two possible approaches.

The first possibility is computing the distance between sentences and features using a classical string metric. In particular, we used the Levenshtein distance, which is a discrete metric (i.e. only takes values in $\mathbb{N}_0$). Heuristically, this metric works by counting the number of single-character edits (insertion, deletion, and substitution) needed to obtain one of the two strings starting from the other (being a well-defined metric, it's symmetric, thus it doesn't matter which is the starting or the ending string) and then defining the distance between the two as the minimum number of such edits needed. An example of how it works can be seen in Fig. 2. More formally, given two strings s_1, s_2 of length $|s_1|$, $|s_2|$ the Levhenstain distance between them can be recursively defined as:

$$lev(s_1,s_2)=\begin{cases} |s_1| & if\ |s_2| = 0 \\ |s_2| & if\ |s_1| = 0 \\ lev(tail(a), tail(b)) & if\ a[0] = b[0] \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & o/w \end{cases} . \quad (4)$$

where $s[i]$ is the i-th character of a string s, and $tail(s)$ is a string composed of all but the 0-th character of the string s. While using a standard metric comes with the benefit of having a defined formula for the computation of the desired distances, without the need for any steps in-between, the usage of this metric comes with the major issue of not being resistant to a change of order of the words composing the sentences. As an example, computing the Levenshtein distance between the two sentences s_1 = $"resolution\ of$ $4K$ and $s_2 = 4k\ resolution$, we would obtain a value of

$$lev(s_1, s_2) = 9 \quad (5)$$

when instead for our purpose the two sentences should be considered similar. To solve this problem, the comparison between sentences and features is performed using the following steps:

- First, all the sentences and features are cleaned from the unnecessary parts; in this step, we remove the stop words and the punctuation, and all the remaining words are stemmed.
- Then, each sentence is iterated through, considering windows of length equal to the max length of any feature and hope length equal to 1; in this context, we define the length of a sentence/feature as the number of words it is composed of.
- For each window, all the possible permutations of the words are considered, and the distance between the window and a feature is computed as the minimum Levenshtein distance between the possible permutations and the features. If the window is at a distance from a feature smaller than a given threshold, we detect a match.

The second possibility is leveraging sentence embedding. Sentence embedding is a general term for techniques that map sentences to real-valued vectors. In particular, sentences that have a similar meaning will be mapped relatively close in the vector space. For this purpose, our model of choice was **all-MiniLM-L6-v2**, a model distilled from Bert that, while with performances comparable to other distilled models of bigger size is approximately five times faster [12]. If the cosine similarity between the encodings of a sentence and a feature is bigger than a certain threshold, it is considered a match. Since the features do not really constitute a complete sentence, to increase the accuracy of the sentence embedding each feature is first cast to a dummy sentence; e.g., the feature "4K resolution" would be cast to the sentence "this product has a 4K resolution".

$$\text{hello} \xrightarrow{\text{deletion}} \text{hell}\cancel{\text{o}} \xrightarrow{\text{substitution}} \text{help}$$

$$\Longrightarrow \text{lev(hello, help)} = 2$$

Figure 2: Example of the algorithm for the Levenshtein distance.

3.3.2 *Audio and video generation.*

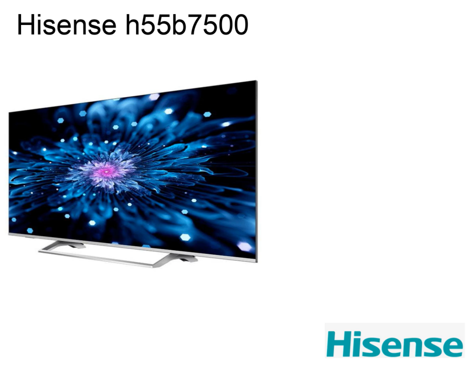
Being now at a sentence-level granularity, the task of generating audio and video to then synchronize becomes straightforward. Together with the dictionary produced in the distance module, the name and the brand of the product are also taken as input.

Now, the **PIL** library is used to generate the slides with the following criteria:

- A first, introductory slide is generated. It contains the image of the product in the sentence, the name of the product as title and the logo of the product in the bottom-right corner. The image of the product will be later used for the template of the following slides. The two images are scraped using *Bing Image Downloader* which is a Python library used to download images from Bing.com.
- For each paragraph, a new slide is generated, which has the title of the paragraph as title and the features contained in the sentences of the paragraphs appearing at the same rate as the associated sentence is read.

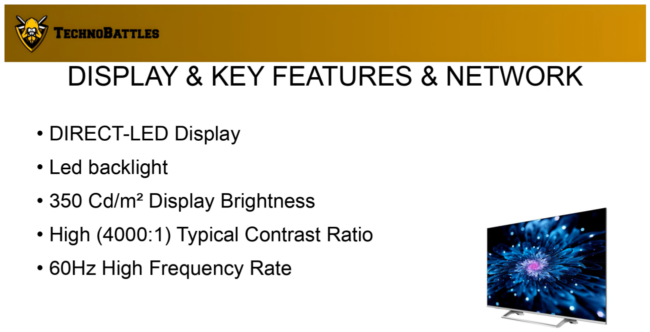
At the same time, the google text-to-speech (gtts) library is used to generate the audio. By iterating through the dictionary, it is clear that the synchronization between the video and audio is easily performed. Once the videos and the audio for all the different paragraphs are generated, they're merged into a final video using the moviepy library.

Some additional details are introduced, so as to make the video more appealing to an eventual consumer: background music is added to the video and a suitable graphic is used for the template of the slides. An example of an introductory and a general slide can be seen in Figures 3, 4.


Hisense h55b7500

Hisense

Figure 3: Example of introductory slide



TECHNOBATTLES

DISPLAY & KEY FEATURES & NETWORK

- DIRECT-LED Display
- Led backlight
- 350 Cd/m² Display Brightness
- High (4000:1) Typical Contrast Ratio
- 60Hz High Frequency Rate

Figure 4: Example of a general slide

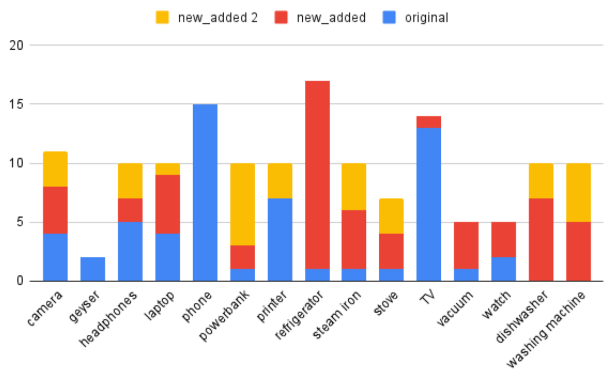


Figure 5: Product categories of training data

4 EXPERIMENTS AND RESULTS

4.1 Dataset

As the core input of the deep learning model, a proper dataset will directly affect the performance of the model. In our project, we used three datasets of different types and structures for our objective:

4.1.1 ***Manually Annotated Product Reviews.*** this dataset is used for training the Customized Named Entity Recognition model, each piece of data is a product review manually tagged with the **product name** ('PROD') and **attribute name** ('ATTR') tags.

We collected these reviews from major e-commerce sites like Walmart, Amazon, and MediaWorld. In particular, since our objective is focused on generating reviews for electronic devices (especially for the display screen, like TV, and monitor...), the product reviews we collected all refer to electronic devices.

As Avignone's [1] work shows, the model does not show a very reliable performance since the training data is not adequate (58 reviews). Hence, the first task we face is to extend the size of the training dataset. During the project, we performed two extensions on the original dataset, as Fig 5 shows.

- **Version 1**: in this version, we added 57 new reviews, introducing new product categories. This new version of the NER model works better on product name recognition, but with a slight performance drop on attribute name tags.
- **Version 2**: considering the possible factor for such decrease, we identified its cause in the difference in tagging style between ours and the original data. So we re-tagged the original dataset to guarantee consistency in the approaches, and meanwhile we added 33 new reviews into the dataset, to keep product categories basically balanced.

By the end of the project, we expanded the size of the original database by nearly three times (146 reviews). In the new dataset, we have **328 product name tags** and **1697 attribute name tags** and 15 different product categories. On average, a review contains 2.23 product name tags and 11.54 attribute name tags.

After the data processing, we process the training dataset to a dictionary with {overview, [list of positions of entities]} format that can be used to train the NER model.

The results show a satisfactory improvement in the performance of the model, which we discuss in detail in the next section.

4.1.2 ***Structured Product Information.*** For fine-tuning the GPT-2 model, we prepared three different datasets, they are:

- **Amazon Phone**: this dataset collected the **mobile phone information** from Amazon, which contains **9570** records with 11 different categories. For our objective, we just need the **Product name** and **Product description**. However, this dataset has a large amount of missing data, as there is a big number of product descriptions with 'NaN' values, so we had to do some data removal. The Final dataset only contains **3414** records.
- **Croma**: this dataset is the most extensive and indicative. It was derived from the web scraping of Croma, a website dedicated to online shopping in India for technological equipment which offers **15436** records with detailed product descriptions. Especially, the description of smart TV is very useful for our task, but only constitutes a small portion of it (around 1.5%).
- **Icecat**: as an extension of the previous work, we add a dataset collected from Icecat, which is an independent global collection of e-commerce product content and product statistics. This dataset contains **6114** records. Unlike previous datasets, this dataset only involves two types of products: TV (3693 records) and personal PC (tablet or all-in-one PC, 2421 records), which have considerable value for fine-tuning our model. Records are already in the HTML format and present several paragraphs and paragraph themes, which is the same structure that we use for our final generated reviews.

Table 1: NER module performance

	ATTRIBUTE			PRODUCT		
	baseline	version1	**version2**	baseline	version1	**version2**
Perfect match	54.25 ± 9.31	48.42 ± 7.79	**56.57 ± 3.97**	56.70 ± 28.91	69.23 ± 13.64	**74.61 ± 7.92**
Partial match	25.99 ± 8.55	26.79 ± 3.35	**24.27 ± 3.17**	30.88 ± 29.13	20.69 ± 7.66	13.44 ± 6.40
Miss	29.22 ± 8.17	31.82 ± 6.52	31.72 ± 5.61	28.31 ± 17.06	24.59 ± 14.89	**8.65 ± 4.21**
Misclassified	7.93 ± 6.84	6.90 ± 5.69	**1.86 ± 0.53**	11.31 ± 11.67	14.28 ± 8.39	**9.79 ± 5.98**
Total match	80.24 ± 8.94	75.21 ± 6.00	**80.90 ± 3.59**	87.58 ± 29.02	89.92 ± 11.06	**88.00 ± 7.19**

Table 2: Special Tokens Setting

Token	Description
OVERV_START	The beginning of a sequence
OVERV_END	The end of a sequence
NAME_START	The start of product title.
NAME_END	The end of product title.
FEAT_START	The beginning of the features list.
FEAT_END	The end of the features list.
NEXT_FEAT	To separate one feature from another.
DESCR_START	The beginning of the description.
DESCR_END	The end of the description.

Our final dataset is the merging of these three datasets, with some preprocessing, eventually containing **24792** records.

However, only using the product name and reviews is not suitable for fine-tuning the GPT model, so data post-processing is necessary. We leverage a previously trained NER model to extract all possible attribute names in product reviews, combined with special tokens, to construct a dataset suitable for fine-tuning the GPT-2 model. Table 2 shows the meaning and configuration of special tokens.

Special tokens can make it easier for the GPT-2 model to understand the training data and avoid being off-topic when generating text.

4.1.3 ***Product Catalog.*** in the generation phase, the data we used is extracted from a product catalog, the catalog is obtained from Icecat and preprocessed to a uniform data structure. Each document contains basic information about the product, such as brand, model, category, and additional specs depending on each product. The product catalog is stored in a dedicated **MongoDB Atlas database**.

In this dataset, most of the products are TVs (28%) and PC monitors (35%). It contains 3 different sub-datasets: features catalogs, features, and items. Since the item dataset includes an exhaustive list of product features, we will focus on this sub-dataset.

During the generation, we will first use as input the product ID as input and, after the search and aggregation operations, we will retrieve a list of product attribute. However, because the list is at times too exhaustive, some features will be useless and even cause a negative effect on the performance of the model. We have a blacklist for this situation which can filter the useless features. Meanwhile, we also prepare a whitelist for the specific products to keep the key features.

In the end, we combine the feature list and product name with the special tokens that can be seen in Table 2, (except the description tokens). We then use this as input to a text generation model to get our desired product reviews.

4.2 Customized NER

In this part of the experiment, we will focus on the Customized Named Entity Recognition model. To train it, all the annotated reviews are used. Rather than loading the pre-existing model present in spaCy, we found it more effective to start with an English blank model to deal with the pipeline component focusing only on the NER. The configuration of the parameters for the training and their respective values are shown in Table 3:

Table 3: customized NER system: training configuration

Parameter	*value*
iterations	100
drop-out rate	0.5
optimizer	SGD

The evaluation is performed using the k-fold method which consists of splitting the dataset into k splits, and for each split, we consider it as the test set for the NER module trained on the remaining splits. The evaluation metrics used to assess the performance are the followings, using k-fold:

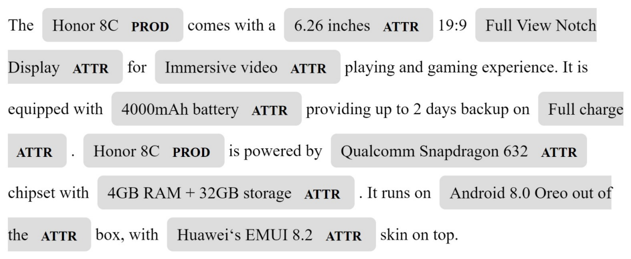
- **Perfect match rate**: it indicates those attributes and product names that are exactly retrieved with respect to the whole set of detected elements.
- **Partial match rate**: it counts the detected entities with partial overlapping only, thus not reporting the complete original element
- **Missed rate**: it provides the rate of entities that aren't even partially detected from the reference set
- **Misclassified rate**: it corresponds to the rate of both perfect and partial matches which are however assigned to the incorrect tag

- **Total match rate**: it measures the total system ability to correctly detect entities when both perfect and partial matches are considered valid

The baseline provided by [1] can be seen in Table 1 respectively for attribute and product tagging.

In the following section, we will show the steps we have performed to create the final version of the model. An intuitive approach to follow in order to increase the performance would be to enlarge the dataset size. For this reason, we almost triplicated the manually tagged dataset and created a new NER module based on the new training dataset. This specific version will be denominated as $ver1$. An important thing to notice is that, despite having an increase in performance with respect to product tagging, the performance was not outstanding, especially for attribute performance which also saw a decrease in performance. This was because the originally provided dataset used a different way of tagging. After all, it is a very subjective aspect: some words may sound like attributes to a person but may not for others. For this reason, a further step has been done which is to adapt the original 56 reviews present in the original dataset to our specific tagging, thus obtaining $ver2$. In table 1, we report the evaluation of all the versions of the NER module and highlight the most important improvement we have done so far.

Table 1 shows how the last version of the module performs overall better in terms of variance. Related to the mean value instead, we notice a huge increase in performance in Product detection, in particular, related to Perfect match, Missed, and Misclassified, meanwhile for the attribute we can point out the significant improvement in misclassification. A simple application of the NER module can be shown in figure 6



The Honor 8C PROD comes with a 6.26 inches ATTR 19:9 Full View Notch Display ATTR for Immersive video ATTR playing and gaming experience. It is equipped with 4000mAh battery ATTR providing up to 2 days backup on Full charge ATTR . Honor 8C PROD is powered by Qualcomm Snapdragon 632 ATTR chipset with 4GB RAM + 32GB storage ATTR . It runs on Android 8.0 Oreo out of the ATTR box, with Huawei‘s EMUI 8.2 ATTR skin on top.

Figure 6: Tagging example of the customized NER module

The best-performing model will be used to generate the training input for the GPT-2 fine-tuning, as explained in the Methodology section.

4.3 Text Generation Module

The natural language generation model we have used for our experiment is GPT-2, which is regarded as one of the best-performing in the field of NLG. Since the default GPT-2 provided by Open-AI is trained to generate very general text, it is not suitable for our purpose. For this reason, fine-tuning of the model is required. For the fine-tuning, we used the final dataset collected from 3 different databases. Notice that different databases contain different fields with different features, so to extract the necessary part, we keep only the product name and the description of the product. We notice that by passing only the product name to the GPT-2 as input, the generated review is coherent with the given task, including technical information extracted from the technological field. However, the mentioned attributes are hardly consistent with the specific category. For this reason, we use the NER module which helps us identify the specific features of each product from its descriptions. The Transformers library provides the necessary tools for customizing text generation according to the desired decoding technique. Passing as input both the product name and its attributes, the module produces reviews suitable for our purpose. For fine-tuning the original GPT-2 module, the following parameters have been set, in order to properly balance the management of computational resources and the quality of the final model.

Table 4: GPT-2 system: fine-tuning configuration

Parameter	*value*
Module	GPT-2
epochs	5
train-batch-size	4
warm-up steps	500
weight decay	0.01

Under our computation availability, we could fine-tune the GPT-2 medium model, which will be used to compare with the results obtained by [1] which only uses the small model. After fine-tuning the model, some crucial choices have to be done to generate and evaluate the text. Some parameters play important roles in text generation. In particular, we can point out 3 main parameters:

- **Top-k**: given the conditional probability distribution related to the selection of the next term based on the previous terms, the TopK allows us to select only the k most probable words which to choose from, where k is a parameter to be set. When k is small, the model tends to be more repetitive and has less creativity since it will always pick the most relevant terms. Instead, for large k, we may have not readable phrases since the terms may be not very correlated with each other.
- **Top-p**: given the conditional probability distribution, the TopP selects the most probable words until their cumulative probability exceeds p, where p is also a parameter to be set. The same reasoning goes here but with the opposite effect: for large p we may have uncorrelated terms, instead for small p we may have always the same word, generating a repetitive text.
- **Temperature**: as mentioned in section III, the temperature plays a crucial role in generating the review. In particular, when the temperature is low, the model will probably output the most correct text, but rather boring, with small variation. In the opposite case, if the temperature is high, the generated text will be more diverse, however, there is a higher possibility of grammar mistakes and the generation of nonsense.

The parameters used to generate the text in order to compare the model performance are the following:

Table 5: GPT-2 system: text generation

Parameter	*value*
Module	GPT-2 Medium
Temperature	0.75
top-k	35
top-p	0.95

In order to assess the performance of the produced reviews, we use 2 types of metrics:

- *Similarity scores*: they consist of computing some statistics based on the comparison between the generated review and the product description. In particular, we will consider the two:
  - **BLEU** [4][6]: BLEU score is obtained by comparing the generated text and the reference one, and consists in computing the n-gram overlapping (geometric average precision score) and also taking into consideration the brevity penalty. The final score ranges between 0 (completely uncorrelated texts) and 1 (identical texts).
  - **GLEU** [5]: GLEU works similarly to BLEU, but it takes the minimum between the precision and the recall of the overlapping n-grams. The GLEU score ranges between 0 (no matches) and 1 (all matches) and it is symmetrical when switching output and target.
- *Syntactical scores*: they are computed based on the syntactical properties of the generated review. From [6] we extract three notable metrics for the evaluation:
  - **Flesch Reading Ease Score** [7]: this score ranges between 1 and 100 and suggests which level of education is needed to be able to understand an English text. A higher value means that a lower level of education is necessary and a lower score means that a higher level of education is needed. The computation of the score is as follows:

$$FRES = a - b(\frac{totwords}{totsentence}) - c(\frac{totsyllables}{totwords}) \tag{6}$$

  where a, b, and c are constants and in particular

$$a = 206.385$$
$$b = 1.015$$
$$c = 84.6$$

  A score between 60-70 means that the text is written in plain English [8].
  - **Coleman-Liau Index** [9]: this metric is also used to compute the grade level necessary to read a text. The index ranges between 6 and 19, suggesting the education level required (higher the index, higher the education level). It can be computed as follows:

$$CLI = 0.05888L - 0.269S - 15.8$$

  where L represents the average number of letters per 100 words and S represents the average number of sentences per 100 words.
  - **Gunning Fog Index** [10]: this readability test is used to evaluate the text of about 100 words, so not for an entire text but for paragraphs. It takes into account the number of complex words. A word is complex if it consists of three or more syllables and it is not a proper noun. The index estimates the years of formal education a person needs to understand the text at the first reading and ranges between 6 and 21. The index is computed as follows:

$$GFI = 0.4[(\frac{words}{sentences}) + 100(\frac{complex - words}{words})] \tag{7}$$

In order to have a comparison between the models, we use the same parameters used by [1] to generate the reviews, in order to compare the advantages provided by the GPT-2 medium model. In addition, we will also provide statistics related to the Syntactical score both for the small and medium model, since those scores were not introduced in[1].

Table 6: GPT-2 text generation: evaluation

metrics	baseline*	GPT2-medium
BLEU	0.467	**0.529**
GLEU	0.52	**0.577**
Flesch	46.66	**50.05**
Coleman-Liau	18.13	**17.6**
Gunning-Fog	14.96	**14.23**

1

From Table 6 we can notice that using the GPT-2 medium gives overall improvement with respect to the small version. We also notice that there is improvement in the overall system by using the improved version of the NER to pre-process the provided dataset.

- *Remark*: notice that the test performed here provides only a general evaluation between the 2 models. During the real usage of the system, we should pay particular attention to how to use the appropriate parameter to assess the requirements of our end users. In particular, depending on the level of knowledge in the technology field of the end user related to the product, we may generate different texts. For instance, the temperature of the generation module may be set according to certain criteria since a high level of temperature (higher than 1) may generate more diverse and readable text but it may contain less specification. Instead, a text generated with a low level of temperature is more specific in terms of the specifications of the product but may sound less fluent. Another particularity to point out is that those evaluation parameters just give an idea of the possible performance of the module. The best evaluation method possible is still human evaluation, since totally different texts may have the same meaning which cannot be taken into consideration by the used scores.

4.4 Video generation

To discuss the experimental results obtained by the Video Generation module, we will again consider separately the distance module and the video and audio generation module

1*notice that baseline performance is based on preprocessing the database using the baseline NER system

4.4.1 Distance module. As we discussed in the Methodology section, in this module we considered two different approaches: using the Levenshtein distance and using a sentence embedding. To understand the differences that come from the utilization of the two, we will now define two types of matches between a feature and a sentence:

- **perfect match**: a feature and a sentence are considered a perfect match if and only if the feature is explicitly mentioned in the sentence, regardless of the ordering of the words. E.g., the sentence "this product has a resolution of 4K" would be considered a perfect match both with "resolution of 4K" and "4K resolution".
- **semantic match**: a feature and a sentence are considered a semantic match if and only if the sentence has a meaning similar to the one expressed by the features. Notice that such similarity is to be considered a subjective quantity. For E.g. the sentence "this product has a resolution of 4K" may be considered a semantic match to the feature "High resolution", and to some extent to "great viewing experience" as well.

First, we underline how the amount of matches is strictly dependent on the threshold chosen for the similarity, especially for what regards the sentence embedding. Under our configuration, we have a drop in perfect matches from the Levenshtein distance to the sentence embedding of approximately 7.9%. However, the main difference between the two comes from the possibility of having semantic matches when using sentence embedding. Indeed, we found several instances of it happening, leading to some interesting matches between sentences and attributes, e.g.:

- slim $\longrightarrow$ flat screen
- greater viewing area $\longrightarrow$ 65 inches
- remote is easy to operate and the remote button is easy to press $\longrightarrow$ remote one-touch access

Clearly, whether the possibility of a semantic match is desired or not is a design choice. Given the scope of this project, we eventually deemed the semantic match to be useful.

4.4.2 Audio and video generation. since no metrics exist to evaluate the quality of the final slideshows, we needed to use an alternative approach. In particular, starting from a certain "id" associated with a product, we tried to evaluate the quality of the relative slideshow just by watching the generated video. We found that the results are quite promising. However, there could be cases in which anomalies come out like when two or more features are repeated in the same paragraph or when the generated textual review presents some inaccuracies. Since there is not central database for the product images, we decided to homogenize their research by leveraging, as we already said previously, a Bing downloader. The API does not provide many functionalities so it could happen that rarely the image for the product may not be the right one.

5 DISCUSSIONS AND FINDINGS

In a context in which the rush to consume is increasingly insistent [11] we think that our work could be a good basis for a solution that may help. With the advent of increasingly advanced technological products, the ability to facilitate the search for appropriate products could have a significant impact on society. In this work, we presented a framework that can be used to generate video / audio reviews of technological products starting from a list of features and capabilities to address a wide range of users. The capability of tailoring the review based on the typology of the user was one of the core points of our project since the very first phases. We are convinced that this can be the key to the final users' satisfaction. Increasing the database used to train the NER and adopting the GPT-2 medium helped to increase the performances and obtain results more and more promising. Much still needs to be done in order to increase the quality of the final solution. For example, one could reason about switching to a NER based on BERT instead of leveraging Spacy, making the GPT-2 more robust in different domains, adapting the generated reviews to more disparate groups of people, and at the end making the final video more humanized. Moreover, since the quality of the video review is heavily dependent on the quality of the generated textual review, which in turn is dependent on the quality of the considered dataset, a huge improvement would come from the utilization of a more suitable dataset. However, there exist no datasets already containing the information needed for our task. Indeed, matching reviews to the features they contain is a lengthy task that would require huge resources to be accomplished in a large scale, making the utilization of a NER (and the error that is therefore introduced) necessary.

REFERENCES

[1] Andrea Avignone, Generation of product descriptions based on structured data using an NLG approach. Rel. Alessandro Fiori. Politecnico di Torino, Corso di laurea magistrale in ICT For Smart Societies (ICT Per La Società Del Futuro), 2022

[2] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.

[3] Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

[4] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. doi: 10. 3115 / 1073083. 1073135. URL: https://aclanthology.org/P02-1040 (cit. on p. 73).

[5] Yonghui Wu et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. 2016. doi: 10.48550/ARXIV. 1609.08144. url: https://arxiv.org/abs/1609.08144 (cit. on p. 73).

[6] Giuseppe Rizzo, Thai Hao Marco Van. Adversarial Text Generation with Context Adapted Global Knowledge and a Self-attentive Discriminator.

[7] F. Rudolf, A new readability yardstick, Journal of Applied Psychology 35 (1948) 333–337.

[8] F. Rudolf, How to Write Plain English, Barnes & Noble, 1981

[9] . Coleman, T. L. Liau, A computer readability formula designed for machine scoring., Journal of Applied Psychology 60 (1975) 283–284.

[10] J. Bogert, In defense of the fog index, The Bulletin of the Association for Business Communication 48 (2) (1985) 9–12

[11] Arıkan Saltık, Işıl & Firat, Aytekin & Kutucuoğlu, Kemal & Tunçel, Özgür. (2013). Consumption, consumer culture and consumer society. Journal of Community Positive Practices. 13. 182-203.

[12] Nils Reimers, Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, EMNLP 2019.