Master research Internship

Internship report

Deep Multimodal Embedding for Image-Repurposing Detection

Domains: Machine Learning - Multimedia

Author:
Antoine CHAFFIN

Supervisors:
Ewa KIJAK
Vincent CLAVEAU
LinkMedia, IRISA

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Abstract: With the growth of social networks, we observe that information media people choose are changing. Rather than getting information from conventional media outlets, more and more internet users choose to stay informed through Social Networks (SN). However, these SN posts are seldom written by professionals but by anybody and with few indications on the ethics or agenda of the poster, thus, these pieces of information can be false, purposely or not. This is called fake news. One type of fake news is named image repurposing. It consists in altering one part of a multimedia publication such as the caption (because it contains most of the information) while keeping the image untampered to misrepresent or repurpose the image. For example linking the picture of an old hurricane to an actual hurricane results in misleading readers. The tremendous amount of information being posted at a time coupled with the quick propagation of information that became viral create a need to build automatic tools that help the user to know if a post is fake or not. While progress has been made in tasks using only one type of media, combining two or more types is still challenging, this has led to a relatively small number of studies being published in the image repurposing detection field. Indeed, making a model able to detect such fakes needs to deal with continuous representation of both textual and image information. These representations are used to assert the integrity of the publication by verifying that both share the same semantics. This assertion is also often done by comparing it with other verified publications in order to check if the publication is not a modified version. A collection of such publications is often called a Knowledge Base (KB) and building one is also a fairly new task, resulting in a small amount of KB which is disadvantageous both for training models and evaluating them.

Contents

Acknowledgements

I would like to thank both of my supervisors, Ewa KIJAK and Vincent CLAVEAU, for their help, their advice and their leads that made this work possible.
I also would like to thank my classmates: Taha, Enzo, Julien, Gaël, Olivier and Willou for their support and our very interesting conversations during the year.

1 Introduction

1.1 Fake news and image repurposing

Following the growth of social networks, internet users are starting to use these platforms as sources of information instead of using conventional news outlets. This is especially true for young people as shown in [1]. However, when it comes to information, social networks have two negative aspects leading to the propagation of erroneous information.

Firstly, anyone can post anything, independently of level of expertise or candidness of intention. Thus, false information appears (often called fake news) which then may be widely shared and may become viral. The spreading of such information may be due to a desire to manipulate (for political, commercial or other reasons) or due to simply poor knowledge of the subject. Secondly, the volume of information circulating on social networks is considerably bigger than on traditional media outlets, meaning that analyzing this information is comparatively complex and costly. Besides, there is the considerable speed at which information that has become viral spread. For these two reasons, it is desirable to implement automatic tools for detecting fake news, in order to be able to mark them as fake as quickly as possible, since the manual verification of each piece of information is far too long and costly.

There are many types of fake news, ranging from digital editing of an image to deceive the reader (e.g. adding a third arm on photoshop), to decontextualization of speeches to deceive the reader on what the speaker meant. In this study, we will focus only on one specific type of fake news called image repurposing. This one is based on the multimedia aspect of information on the internet. Indeed, images are often uploaded with a caption and a set of metadata such as time and location of the shot. This combination of visual information and textual/contextual information is called a multimedia **package**. Each data collection taken by the same acquisition tool (CMOS sensor, microphone, keyboard, etc.) is called a modality.

The principle of image repurposing is to alter one of the modalities while keeping the visual modality intact. Typically, it consists in re-using a previously posted image without altering it but modifying, for example, its description, in order to convey false information. For example, in figure 1, a picture taken after a festival representing a park littered with garbage has been repurposed to make the reader think that an environmental manifestation is at the origin of this pollution in order to discredit the movement. Similarly, when hurricanes occur, image repurposing is commonplace as users upload pictures of other hurricanes, often making the situation look worse than it is and causing panic.

It should be noted that this study focuses only on unaltered images, as image alteration is a well-studied subject and it can be assumed that the model will work downstream of an image tampering detection module, in order to provide it only with unaltered images.

To detect image repurposing, textual and image information need to be compared to ensure they are semantically and contextually related. Moreover, a package to be verified (in this case called a query package) can be compared to unmodified packages contained in a collection that is called a **Knowledge Base** (KB) to verify that it is not a modification of an existing package in the collection. In order to assert the semantic integrity and compare packages, the information needs to be correctly represented so it can be correctly processed by modern deep-learning models such as neural networks.

Figure 1: Example of image repurposing. The original picture has been taken after the 420 festival, yet the author claims that it has been taken after an environmental manifestation in order to discredit the movement.

1.2 Modalities representations

In the recent years, there has been a sharp increase in multimedia-related problem-solving using deep learning methods linked to both the increase in the computing capacity and the creation of large databases. However, these databases are not directly exploitable by standard neural networks: these networks take compact and continuous representations of information as inputs whereas data from original domains (e.g. image, text, ...) is not in this form.

It is therefore needed to extract a continuous and compact representation of the data, that represents its characteristics like its semantic context. Such a representation should have interesting properties, including notion of distance, that is, a low distance between representations of similar data and a large one for dissimilar data. This representation can be a conceptual representation, i.e. a vector marking the presence or absence of certain concepts understandable by humans. In this case, it is said that it belongs to the concept space. It can also be more abstract, symbolizing characteristics that cannot be interpreted by humans and that are extracted by a neural network to perform a specific task. In this case, it is said it belongs to the representation space.

In the case of image repurposing, the representation of the image and the representation of the text (or other metadata in the package) need to be combined, either by making a joint representation or by processing them separately but for a common purpose.

1.3 Objectives

Objectives of the internship are exploratory. One of them is to study the possibility to create realistic datasets in an automatic fashion, another is to study if state-of-the-art multimodal models can be applied sucessfully to the image repurposing task and the last is to explore how such models' decisions can be explained.

This report will first introduce techniques used to obtain continuous representations of each modality individually, focusing on image and text only, because together they contain most of the information, as well as techniques used to build or process these representations together. The state-of-the-art of image repurposing will then be introduced through the study of different methods implemented, followed by an analysis of the evaluation of presented models. Next, researches done during the internship will be presented by detailing, for each objective, proposed methodologies and a study of obtained results. Finally, a summary of the remaining issues in the field and leads to explore to solve them will be presented.

2 Modalities embedding

Conventional machine-learning techniques are not built to deal with data in their raw form: they work better with a continuous representation of modalities which represent features of the raw data and from which the learning system will be able to detect patterns. These representations are often in the form of vectors and are called distributed representations or **embeddings**. To form a representation of both modalities, one can represent each modality separately and then merge them to get a joint representation, i.e. one vector for both modalities or use a translation to get a representation in the same space of the two modalities, one vector for each.

While in the past relevant features were mostly selected "by hand" via a study of experts in the field, the current trend is the automatic extraction of features through deep-learning techniques. Such techniques allow representation-learning with multiple levels of representation obtained by the composition of the representation of the prior layers, starting from the raw input. Each layer then extracts a higher, more abstract representation. In addition to matching the format of the expected input, these embeddings should correctly describe the object (such as the meaning for a word) and have relevant properties, like small distance between two vectors describing two similar words (in their spelling or their meaning) and a large one if they are not similar at all.

2.1 Image embedding

Convolutional Neural Networks [2] (CNN) are an adapted version of standard neural networks that can take an image as input by embedding it in a Euclidean space which serves as a basis to perform the intended task. The way an image is translated into a continuous vector can be compared to the human visual system.

Indeed, when we see an image, the eyes extract information from different visual regions, forming an electrical signal if certain patterns are detected in certain regions. These signals can be seen as really low-level features in the field of view and these features are then successively combined in different parts of the brain in order to form a sufficiently abstracted representation, understandable by other parts of the brain. Similarly, these processes can be adapted in the form of algorithms used by computers.

In order to model the activation of visual regions, the concept of convolution is used. This process is equivalent to applying sliding filters to the whole image and allows to extract local information describing the image[1]. These filters are learned during training and are activated when a discriminative pattern that makes it easy to perform the intended task is found in an image. The convolution can be seen as a feature extractor that finds invariant features so the model can detect an object no matter how and where it is located in an image. These patterns are then combined through a succession of hidden layers in the neural network, leading to more precise detections. These hidden layers are convoluting the previous layer in order to detect local conjunctions of features from the previous layer and then pool it (i.e. synthesize multiple parts of the input into one) to reduce the possible variance (small variations) in the input, leading to a more abstract representation of the previous layer.

Starting from an array of pixel values, the first layer will, for example, detect the presence or absence of edge with particular location and orientation, whereas the second will detect particular arrangements of these edges. The next will assemble these arrangements to form parts of familiar objects and subsequent layers will combine these parts to form the actual objects. This is illustrated in figure 2. Again, these combinations are learned during training to find discriminative patterns.

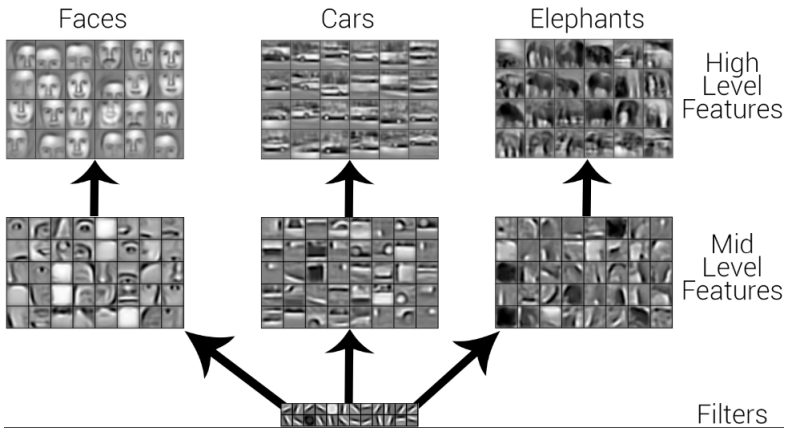


Figure 2: Representation of filters and different levels of features extracted for a detection task, [3]

CNN became state-of-the art in many computer-vision tasks after VGG [4] made a huge improvement in the classification task. This improvement also led to advancements in the representation of images, given the effective feature extraction done by these models. Intermediate layer activation values are now used to as an embedding of an input, which represent its features at different levels of abstraction, depending of the depth of the layer.

From there, researchers found that a more accurate model on a given task leads to a more precise representation of the input in intermediate layers. It follows that building more accurate models leads to more accurate embeddings. For a long time, the creation of more accurate CNN-based models involved stacking more and more hidden layers, allowing the network to create complex patterns that are highly discriminatory and therefore, give a more representative embedding. This approach, however, has been limited due to problems in the training of such deep networks because of what is called "vanishing gradient". The more layers the gradient goes through, the lower the

[1]For more details on convolution, reader can refer to: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

gradient is because of saturating units (high output using certain activation functions results in very small derivative and small weight updates). The consequence is that early layers will barely be updated leading to poor training and poor results.

There are several ways to deal with vanishing gradient, the easiest one being to use the ReLU activation function which has a high derivative even with high output. Another way is to use skip-connections, that is, adding the output of the previous layer to the output of the following layer, so the unit only learns a mapping of the residual value instead of the real output. If we denote the real output by y and the input by x, instead of learning $y = f(x)$, it learns $y = f(x) + x$, with $f(x)$ being significantly lower, which is easier for a network to learn because it does not saturate. Also, the gradient can directly backpropagate through these identity layers, allowing earlier layers to be updated even in a deep network. The ResNet model [5] is built using skip-connections and has increased state-of-the-art results leading to really precise embeddings. Yet, these architectures are really deep are therefore very demanding in terms of computing power, so models with less accurate results but much lighter in the implementation are often used.

Finally, these networks can also be trained in an unsupervised manner. Indeed, while a network can be trained for a supervised task such as classification with an annotated dataset, one can also train it using autoencoders [6]. An autoencoder is composed of two parts, the first one, the encoder, synthesizes the input which is in the original domain into a representation space by stacking hidden layers with a decreasing number of units. The second one, the decoder, synthesizes back a vector from the representation space into the original space. So the network is trained to reconstruct the input as closely as possible after casting it to the representation space. A representation can then be extracted the same way as a network trained with supervision, by taking activation values in intermediate layers. This training is called unsupervised training because the dataset is composed only of non-annotated images, thus, making the creation of a training dataset much easier.

2.2 Text embedding

Text can have a variable length: it can be individual words, phrases (of variable length), sentences, phrase set, paragraphs or full documents. However, its representation must have a fixed size to be processed by a neural network or even to be compared to another text's representations. Also, text can be represented as a set of independent words or as a sequence of words. Thus leading to two types of embedding, respectively static and dynamic embedding. The first one does not take into account the context of the word to produce its representation while the second one does.

A naive approach to represent a word as a fixed-size vector, said one-hot, is to represent it as a vector with the size of the vocabulary filled with 0 and a 1 at the position of the word in the vocabulary. Yet, this straightforward approach has two downsides. First, this representation does not encode any similarity between similar words, and second, this absence of similarity is exacerbated by the high dimension of the vector leading to the isolation of every word due to the curse of dimensionality.

2.2.1 Word2Vec

In order to get a more compact and meaningful representation, one can, similarly to the processing of images, use the representation extracted by a neural network to perform a specific task, which have a smaller size and encode "features" directly contained by the semantics of the word (which

can be seen as a sort of "meaning features"), allowing to directly compare two different words by studying the similarity of their vector.

Indeed, when training a neural network to perform a task on a corpus by giving it one-hot representations, weights in hidden layers will converge to values that discriminate inputs in order to be able to perform the task. After training, a word can be given as an input of this network and computed values can be obtained at a given layer, which can be seen as a continuous vector representing this word.

In one of the most famous static embedding techniques, Word2Vec [7], the task on which the network is trained is predicting words next to the input word in a corpus, which allows getting a good representation of the "meaning" of the word, given words with similar meanings are often found in the same context (with same words next to them). By giving words contained in a window around the word in a given sentence to the neural network, it is expected to return the one-hot encoded word, this is called Continuous Bag-of-Words (CBOW). One can also train a network to perform the opposite task, that is, predict words contained in a window given the central word: this is called Skip-gram. Diagrams describing these techniques can be found in figure 3. The size of the window can vary, but empirical results have shown that a too-small size leads to unprecise embeddings while a too-large window leads to divergence due to the lack of data. Typically, a window size of three is chosen.

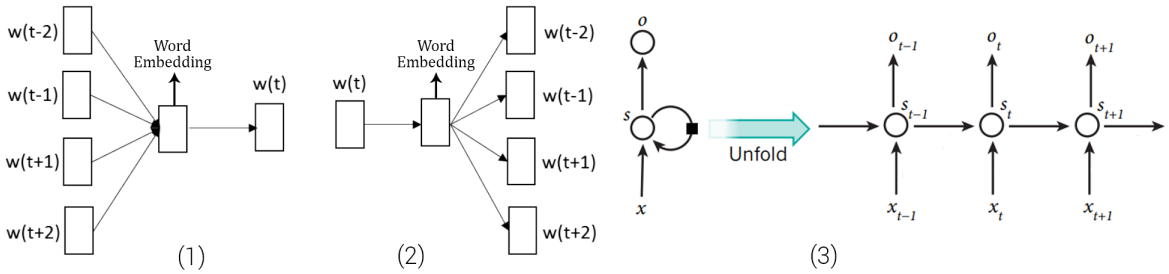


Figure 3: CBOW (1) and Skip-gram tasks (2), [8]
RNN structure (3), [9]

While it can be useful to encode a single word, text is often larger than a single word and can have various lengths, from a sentence to a whole document. A simple way to encode a whole chunk of text is to average the representation obtained for each of its words. While this can lead to a pretty good representation of the text, it does not accurately represent certain properties linked to the text because it does not treat it as a whole. To encode an entire paragraph, a method similar to Word2Vec can be used. The difference here is that the model is trained to predict the next word, given both representations of previous words and the whole sentence. These representations are then iteratively corrected by the network to achieve better results, resulting in better representations.

2.2.2 Recurrent Neural Network and Long-Short Term Memory

Another way to encode the representation of a word given the sequentiality of a text input is the use of a Recurrent Neural Network [10] (RNN). RNN works in the same way as a traditional neural network with the addition of a "state" value as an input. This value represents information about past elements of a sequence. This state is updated after processing each word of the text and used as an additional input to keep an abstract representation of what has been said before in the text

input. One can see RNN as a chain of neural networks sharing their weights, that take, in addition to the next word, the value of the hidden state of the previous network as input, as illustrated in figure 3.

Yet, this method does not work very well for very large inputs because it only accumulates information and never "forgets" it. To correct this, Long Short-Term Memory [11] (LSTM) has been proposed. This architecture provides RNN with an explicit memory that mimics the natural behavior of memory. This memory cell accumulates information, that is, copies its state value and adds to it part of the information from the current input. This value is then multiplied by the value of another unit, the forget gate, which has a value contained between 0 (to forget everything) and 1 (to keep everything) to control the longevity of the memory (this behavior is learned during training). Thus, the network can learn how to manage the content of its memory by adding only part of the information contained in the input and clearing part of it when needed, leading to a significantly more efficient modeling of long-term dependencies. This is really useful when dealing with text because a word can refer to another one relatively far from it in the corpus.

RNN and LSTM activation values can then be used as a distributed representation for a word, similarly as in Word2Vec. In this case, however, the embedding of a whole chunk of text can directly be obtained by feeding the network each word one after the other because hidden states model the whole sentence.

2.3 Attention mechanism

Attention is a mechanism that allow a model to focus on specific region of a feature map, i.e. focus on some vectors of a collection of feature vectors. Its goal is to mimic the human ability to extract the most discriminative information and understand relationships between differents parts of the input. For example, empathize the relation between a pronoun and its subject and adjective in a sentence as shown in figure 4.

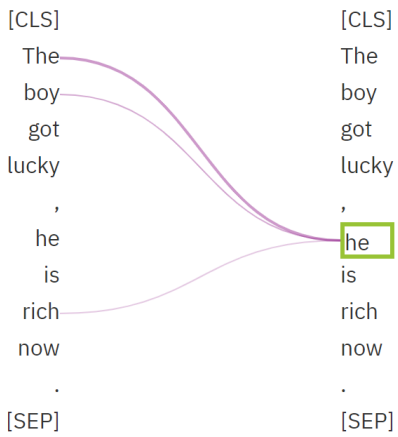


Figure 4: Representation[2] of weights for the word "he" in one attention layer of BERT

It can leads to both improved performance and better interpretability since attention maps calculated show on which features the model focused during the processing. It has already demonstrated its power in improving performances in various applications such as neural machine trans-

[2]http://exbert.net/exBERT.html

lation, speech recognition and sentiment analysis but also in multimodal tasks such as image captionning, video description and cross-modal retrieval.

An attention layer tries to retrieve information from a set of *context* vectors $\{y_j\}$ related to a query vector x. The layer first calculates the maching score a_j between the *query* vector x and each *context* vector y_j. This calculation can either be done using the dot product of the two vectors in the case of multiplicative attention or using a feed-forward network with a single hidden layer in the case of additive attention. These scores are then normalized by a softmax:

$$\begin{aligned} a_j &= \text{score}\left(x, y_j\right) \\ \alpha_j &= \exp\left(a_j\right) / \sum_k \exp\left(a_k\right) \end{aligned} \tag{1}$$

The output of an attention layer is the weighted sum of the context vectors w.r.t the softmax-normalized score:

$$\text{Att}_{X \to Y}\left(x, \{y_j\}\right) = \sum_j \alpha_j y_j \tag{2}$$

In the multimodal field, an attention layer is called a **self-attention layer** when the query vector x is in the set of context vectors $\{y_j\}$ while when the query vector comes from the other modality, the layer is called a **cross-attention layer**.

2.3.1 Dynamic embeddings

Static embeddings obtained by previously described techniques are useful to get a representation of the meaning of a given word. This meaning, however, can depend on the context of the word and sometimes, a word can have a completely different meaning. For example, in sentences: "I know I am right" and "Turn right in 200 meters", the word right has two different meanings. Static embeddings return the same vector for a word, independently of its context, thus, the embedding will be the same for the word "right" in these two sentences. While it is not a problem in certain tasks, the context is necessary to obtain a semantically accurate representation, which is essential in the repurposing detection task.

Dynamic embeddings, on the other hand, take context into account and generate a unique embedding for a given word in a given sentence. In order to do this, two parameters need to be taken into consideration while generating embedding: relative position of the word in the sentence and its relation with other words in the sentence as described in [12].

Modeling the relative position of the word in a sentence is a pretty straightforward task because a vector symbolizing the offset of the word can be added to the embedding vector, leading to a vector representing the word and its position. The second parameter, however, is more complicated to model. Indeed, while it is easy for a human to see the relationship between words in a sentence, this remains a very difficult task for a neural network. It can be done through the mecanism of attention which allows the model to focus on related other words in the input sequence that can help build a better encoding of this word instead of dealing with every other word even weighted. Thus, instead of seeing previous words as a whole in the form of one hidden state, the network can use each word independently and use more related ones to generate a representation of the word. This means that a word is encoded directly using words in the current sentence.

Dynamic embeddings have shown that they are more representative, allowing them to redefine the state-of-the-art in many tasks such as text translation. These representations are indeed more suitable for tasks where context is important, which is the case in image repurposing.

2.3.2 Transformers

The state-of-the-art to generate textual dynamic embeddings is BERT [13] which is based on the transformer architecture. Transformers as first introduced in [12] are composed of an encoding component and a decoding component. Unlike in an autoencoders architecture, the output of the decoding component is not the input of the encoding component but something related such as the next sentence in a corpus or the translation of the sentence. The encoding component is a stack of several encoders and the decoding one is a stack of several decoders. Each decoder gets the output of the last encoder and the previous decoder as input. While every encoder and every decoder share the same structure, each of them have their own weights; an encoder is composed of a self-attention layer followed by a feed forward neural network and a decoder is composed of these two layers with an encoder-decoder attention layer between them, that is, a cross-attention layer between last encoders' output and the previous decoder output. An illustration of the transformer architecture and its components can be found in figure 5.

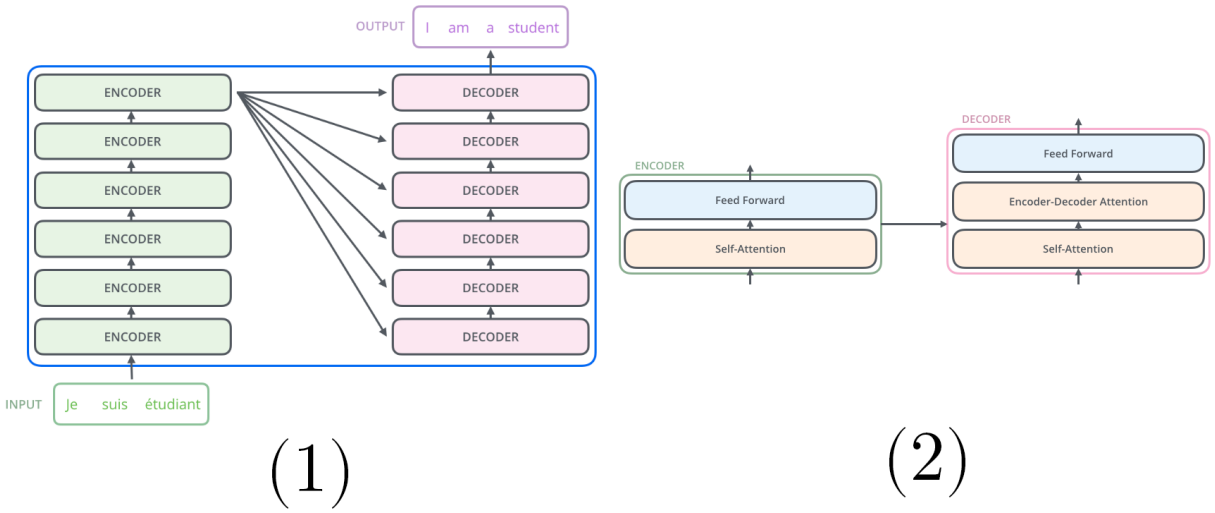


Figure 5: Illustrations[3] of the transformer architecture (1) and its components (2)

An embedding of the input is given as input of the first encoder and then, each output of an encoder is used as the input of the next one. To create even deeper relation and, by extension, better representation of the input, standard attention mecanism is enhanced using **multi-head attention**. Multi-head attention is the process of running several attention layers in parallel, so an attention layer in a transformer is in fact composed of several attention sub-layers which can each create different relationships. This allows more expressivity to the extracted representation.

In a similar way, the output of the final encoder is used as input of the first decoder, which output will be used as the input of the next decoder. In the decoder case, however, an encoder-decoder attention layer is added. This layer uses both the output of the last encoder and the output of the previous decoder to help the decoder focusing on the right parts of the input.

Since BERT's goal is to create a representation of the textual input that can be used for a downstream task, it is only composed of a stack of encoders, discarding the decoding part that

[3] http://jalammar.github.io/illustrated-transformer/

will be designed to suit the task. This type of model is therefore named a transformer encoder as opposed to transformer decoder [14] that are used for generative tasks and are only composed of the decoding part.

2.4 Semi-supervised learning

Most deep learning methods are based on **supervised** learning which requires a large amount of labeled data. The labeling process is often done manually which is both time consuming and expensive when not impossible. Moreover, even in cases where the supervision data is available, this approach creates "expert models" that learned the structure of the training set and are unable to generalize on other tasks or even on data that has a different format.

From this observation, **unsupervised** approaches gained in popularity. A trend in the Natural Langage Processing (NLP) field is to use unsupervised pre-training to find a good initialization point and then **fine-tune** the model (i.e. update its weights) in a supervised fashion on a small dataset to *transfer* the knowledge gained during the pre-training. This *pre-train then transfer* approach is part of the **semi-supervised** category. Its goal is to create a task-agnostic model during the pre-training that will be able to be specialized on various tasks with little changes to the architecture and a small amount of data. The model is in fact learning a *language model*. This approach has been used for years, sometimes without being explicitly mentionned as such. An example is the use of embeddings from Word2Vec in a task-specific model.

Yet, more recently the transformer model brought significant improvements in the expression capacity achievable. Unfortunately, these models are really huge (1.5 billions parameters in GPT-2 [15]) and need a lot of data to be trained. Thus, pre-train these on huge unlabeled datasets and then fine-tune it on small task-specific labeled datasets allow to get very good results while not having to create a big dataset and label it for every task. For example, BERT has been pre-trained to predict a masked word in a sentence. This task is said unsupervised because there is no need for labellisation: given a collection of texts such as the BooksCorpus [16], the creation of a training set by replacing a word by the token [MASK] in a sentence and associate the original word to this sentence is fully automatisable. When trained on this task, encoders are expected to extract an encoding of the masked word in the context of words surrounding it, that is, a dynamic embedding. These embeddings can then be used in a supervised downstream task, thus allowing to transfer what the model learned during the unsupervised pre-training.

This kind of approach is well suited for the image repurposing detection task since the lack of training data is a major problem due to the difficulty in creating a collection of fake news and using unsupervisely pre-trained model could help create powerful fake news detection models even with small datasets.

2.5 Multimodality and crossmodality techniques

In order to combine these two types of embedding, one can use either a multimodal approach or a crossmodal one. The first approach consists in creating a joint representation of both modalities or merge the two. Thus, resulting in a single representation for both modalities but not necessarily in a mapping from the initial representation space to final space or from the final to the initial one. It should be noted that one can just concatenate both modalities's vector but since they lie in different spaces, this representation is not meaningful and is, by extension, underperforming when used in multimodal tasks. The second approach creates the aforementioned mappings and therefore allows

to get a joint representation by translating both modalities from their respective initial space to a common representation space. The crossmodal approach is preferred if it is needed to recover the original domain in the task, like to retrieve a neighbor and then use one of its modalities.

One multimodal technique to get a joint representation of both modalities is to use a Multimodal Auto Encoder [17] (MAE). Monomodal autoencoders can be extended to multimodal inputs by concatenating both modalities in their original space, the expected output is then the concatenation given in the input. One can also create two different networks which both take one modality as input and have a common layer in the middle, which is then used as a joint representation of inputs, the process is the same as a typical autoencoder because the output of each network is just one modality. Both of these architectures are represented in figure 6.

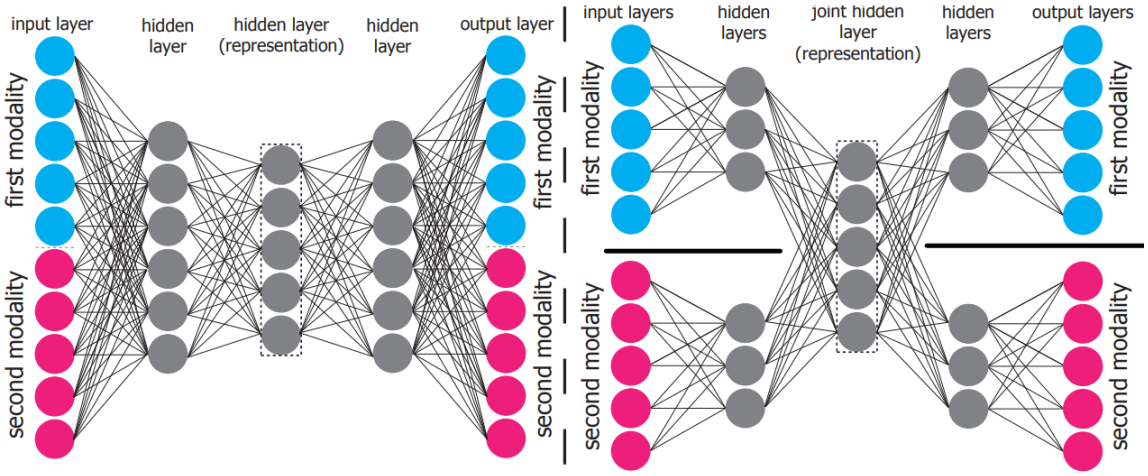


Figure 6: Two types of multimodal autoencoder, [18]

Yet, even in the second configuration, there is a fully connected layer attached to both modalities so the network needs both of them to work. While a joint representation of both modalities can be obtained using this, one can need to map one modality to another or get an embedding of just one modality in the common space. To achieve this, the second configuration can be modified so there is no fully connected layer in the middle but instead, the middle layer is tied by sharing weights between both networks as shown in figure 7. Each network is then trained to translate from one modality to the other by giving each one a modality as input and train it to reconstruct the other one. The weights tied layer can then be used to get a joint representation of both modalities. This configuration, called BiDNN [18], also allows translating one modality to the others' original domain. Moreover, the middle layer can also be used to get an embedding of just one modality, so BiDNN can be seen as a crossmodal approach since one can project both modalities into the same embedding space.

Another crossmodal technique that can be used to represent modalities in the same embedding space is to use space transformation techniques such as the Canonical Correlation Analysis (CCA). Starting from distributed representations of both modalities extracted by previously mentioned techniques in sections 2.1 and 2.2, the goal is to project these vectors into a common space and these projections need to be close to each other if they share the same semantics. This allows the retrieval of similar data no matter the original domain of the input and the retrieved neighbor's one. Minimizing distance between two similar inputs can be done by maximizing the correlation

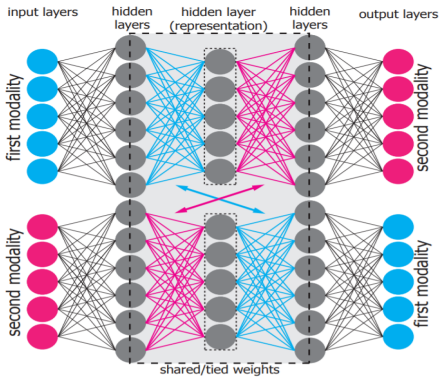


Figure 7: BiDNN, [18]

between the two. If we denote by X the feature vector of the image, Y the textual features and W_x and W_y matrices used to do the projection in the common space, the goal is to find W_x^* and W_y^* such that they maximize the correlation between the projection, i.e.:

$$\left(W_x^*, W_y^*\right) = \underset{W_x, W_y}{\operatorname{argmax}} \operatorname{corr}\left(W_x^T X, W_y^T Y\right) \tag{3}$$

This equation can be extended using the formula of the correlation between two vectors, leading to (with Σ_{yy} and Σxx being covariance matrices and Σxy cross-variance matrix):

$$\begin{aligned} &\max_{W_x, W_y} \operatorname{tr}\left(W_x^T \Sigma_{xy} W_y\right) \\ &\text{s.t: } W_x^T \Sigma_{xx} W_y = W_y^T \Sigma_{yy} W_y = I \end{aligned} \tag{4}$$

Maximizing this relation between every related couple of image-caption leads to the creation of a joint representation space where it is possible to find similar packages using distance or similarity between vectors. A solution to this maximization problem can be obtained through Singular Value Decomposition (SVD) using eigenvalues and eigenvectors of correlation matrices.

3 Image repurposing detection

Image repurposing detection being a fairly recent subject, there are only a few studies on it. There are four of them and they all provide a different methodology with incremental performance. Because every study has a different approach and is trying to compensate for the weaknesses of others, datasets used are not consistent between each study, each new one introducing more challenge in the task.

Most of studies are using a collection of untampered packages that is called a Knowledge Base (KB) and can be used to check if the query package is a modification of an existing package. This type of collection needs to contains as much information as possible in order to make the model robust to every possible package.

It should be noted that the limited data available to train models led studies to find solutions to fill this gap such as the use of transfer-learning by using pre-trained on other tasks models for feature extraction to capitalize on existing datasets or the use of Generative Adversarial Networks [19] to generate synthetic training data.

3.1 Semantic integrity

In the paper [20], authors propose to check the semantic integrity of the image and the text under the assumption that modifying the package would lead to a semantic inconsistency between the two. In order to do this, they first construct a joint embedding of both modalities and then calculate an image-caption consistency score (ICCS). This score is calculated given the assumption that consistent packages follow a certain distribution. If a new package is far from this distribution (i.e. is an outlier), then it is semantically inconsistent.

To build the joint embedding, they use techniques that have been detailed in the previous section such as MAE and BiDNN. Given these representations, one can build a distribution representing clean packages contained in the knowledge base and then study if the package to judge fits this distribution (i.e. is an inlier) or does not (i.e. is an outlier). This judgment can be done by using either a One-Class Support Vector Machine [21] (OCSVM) or an Isolation Forest [22] (iForest). The OCSVM is trained in an unsupervised manner by giving packages in the KB as positive examples so a decision function is learned based on the distribution of these training examples. This function is then used to tell if a new package is indeed part of this distribution. The iForest is a collection of decision trees. Each tree recursively splits the dataset until a node contains only one individual. The number of splits needed to isolate the point is then used as an indicator of its outlierness, indeed, outliers lie in low-density regions compared to inliers so it is easier to isolate them. The training and inference process are illustrated in figure 8.

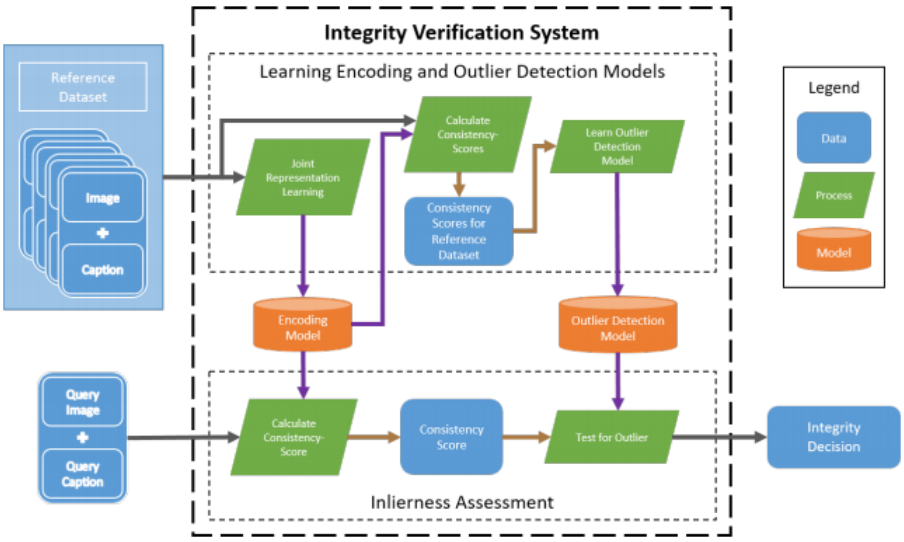


Figure 8: Illustration of the training and inference process of the semantic integrity approach, [20]

Tests have been made by randomly shuffling pairs of image-caption from 3 datasets: Flickr30K[4], MS COCO[5] and one that authors created using pairs downloaded from Flickr (MAIM[6]). While this approach gives good results on randomly shuffled pairs of image-caption, a modification can be semantically consistent and this type of modification is the hardest to detect by humans.

[4]http://shannon.cs.illinois.edu/DenotationGraph/
[5]http://cocodataset.org/
[6]Available on request to the authors

3.2 Package comparison

Given the previous observation that semantic consistency is not sufficient enough to detect intelligent manipulations like the coherent replacement of a type of entity, for example, the replacement of a personality by another personality or one place by another one; authors of [23] showed that it is needed to build both a more robust model and a more challenging test dataset to demonstrate that the previous model is not efficient enough and to evaluate the performance of this new model.

To build this dataset, they first clustered packages by relatedness in order to swap metadata from similar packages only, so there is no modification which is obvious to detect because the resulting package does not make sense. This clustering is first done by GPS coordinates proximity and is then refined by similarity based on image and text features. Then, they applied named entity recognition for different types of entity (person, location and organization) and swapped these entities with one of the same type in a related package thus leading to a semantically consistent and plausible package. This results in the Multimodal Entity Image Repurposing (MEIR[7]) dataset.

For the detection, they start by retrieving related packages in the KB using features' similarity, that is, cosine distance between respective feature vectors. Vectors used in this process come from VGG19 for the image and averaged Word2Vec for the text. They then check the semantic consistency of the package but also check if the package is a manipulation of the top-1 retrieved package after ensuring that they are indeed related. To ensure relatedness, the Jaccard index is used to determine how much the two packages overlap.

The final decision is based on both semantic consistency and package manipulation detection and, if no linked package is found, the second part is omitted leading to the semantic consistency part only being considered. In this way, a package which is not similar to any one contained in the knowledge base is not penalized.

The semantic consistency, like in previous work, is checked only by considering the package itself, without referring to the knowledge base. It is, however, unlike previous work, done by a feed-forward neural network trained to label a package as tampered or not with the package's features as input. A similar network is trained to check if the package is a manipulation of the most similar package but takes the concatenation of both package's features as input. The whole process is illustrated in figure 9.

Results show that, while performance gain is not that significant on datasets used in the previous approach, the model performs significantly better than the previous one on the MEIR dataset, which is more challenging than random shuffling.

3.3 Event verification

Another approach [24] is based on the assumption that an image is linked to a particular event. It consists in extracting features of images and associate these features to the event from which it was taken, resulting in a KB in the form of couples of events and their features. Then, to detect if an image is repurposed, one can extract its features and check if they match features associated with the event it claims to be related.

This study is largely focused on feature extraction, given that the approach is based on the precision of the representation extracted. The authors divided the study into two approaches, one on global image features and the other on local image features. Each built on top of ResNet, using transfer-learning to achieve great results.

[7] https://github.com/Ekraam/MEIR

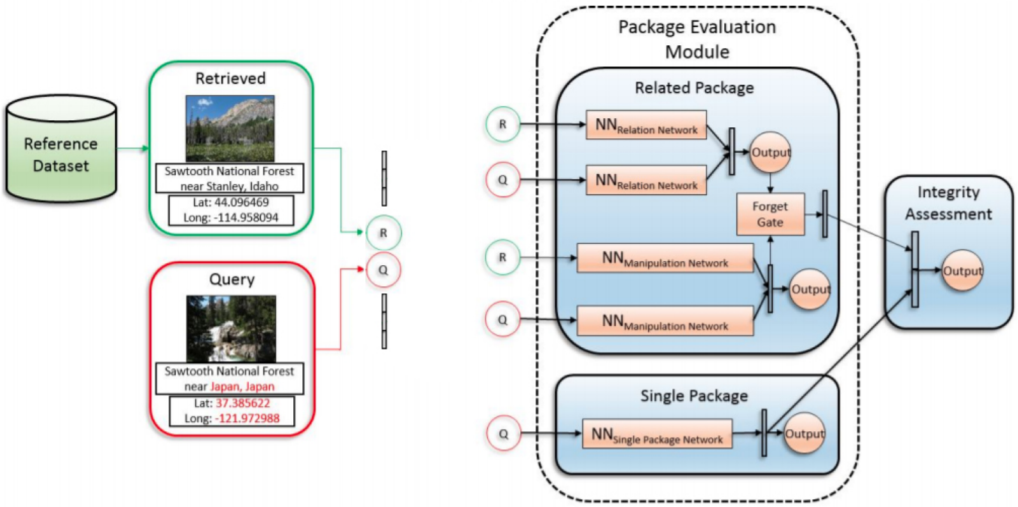


Figure 9: Illustration of the package comparison approach, [23]

The first one maps the entire image to a feature vector using CNN. They used ResNet both with and without fine-tuning, that is, modifying ResNet weights by training it for this specific task or not. Fine-tuning helps to boost performance but needs a pretty big dataset to avoid overfitting on the fine-tuning dataset. Yet, even with a relatively small dataset, fine-tuning the model gave better performance than the untuned model. The second one slices the image in smaller patches and extracts features from these patches using ResNet. The output feature vector is then computed either by summing each vector and normalizing the resulting vector to one or by concatenating every vector. These four different image features extraction methods are illustrated in figure 10. Then, for these two approaches, a classifier is trained to link an image to an event by taking the feature vector of the image as input.

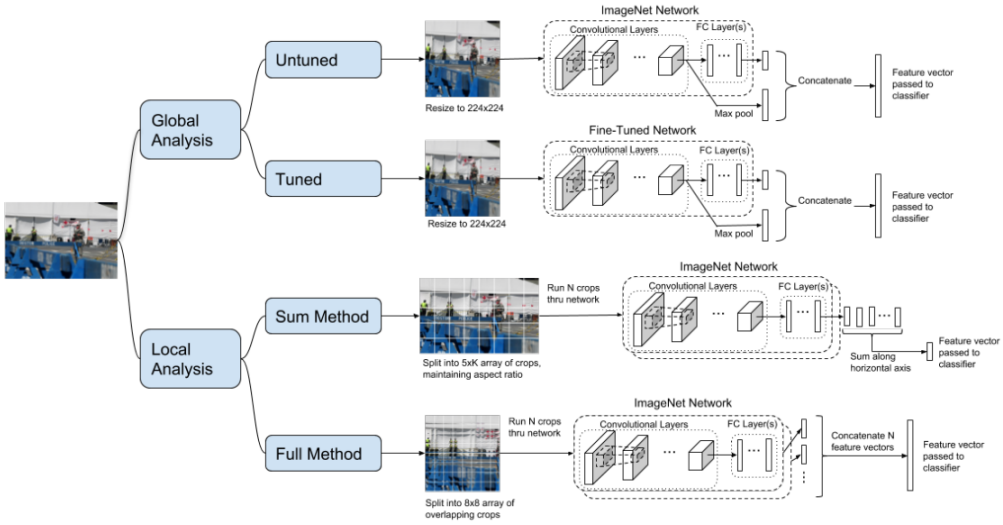


Figure 10: Illustration of the different image features extraction pipelines used for event verification, [24]

Results show that a global approach using the tuned model works the best. Yet, the authors do not compare their approach to other image repurposing detection studies but only used the Event Verification dataset generated by NIST. This is mainly because while event verification could be used to detect tampered packages, it is restricted to event linking. False event linking is a sub-problem of image repurposing and will not perform well on general-purpose, given the lack of data (there is often only one picture for an "event") and the fact that an image may not be linked to any event. However, an in-depth study of feature extraction can be re-used in other approaches.

3.4 Adversarial Learning

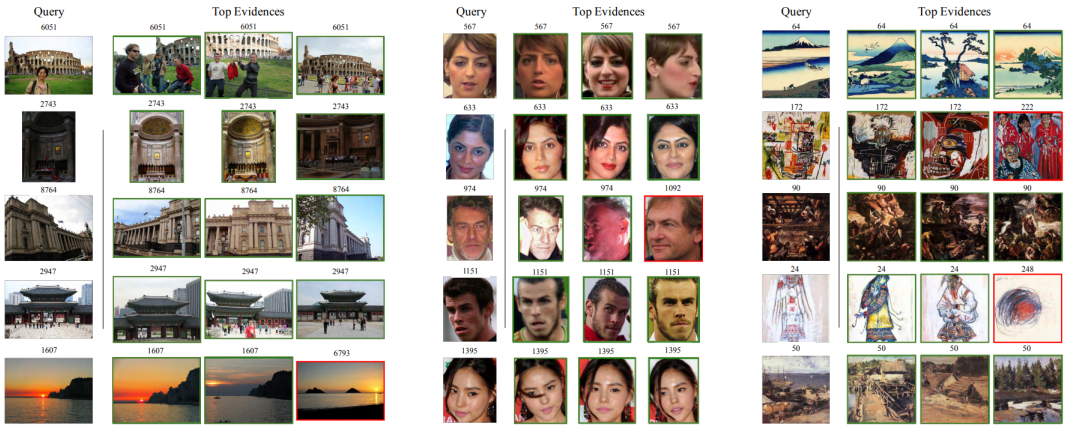
The principle of adversarial learning is to train both a generator and a discriminator at the same time so they both make progress using each others' progress. Thus, this principle is very well suited in the image repurposing detection task, given the lack of (well-formed) malicious data and the need to train a model capable of discriminating fake news from real news. From this, authors of Adversarial Learning Framework for Image Repurposing Detection (AIRD) [25] built a model where the generators' goal is to create convincing fake news by generating a fake caption for an image while the discriminators' is to be able to tell if a package is a fake news or not. This is similar to real-world scenarios where a fact-checker becomes more and more efficient because the faker is producing more credible fakes and vice versa. Fake news created by the generator become so plausible it can then be used as a generator to make a dataset (and can, therefore, be used to train other models) and the discriminator can be used to detect fake news in real-world situations.

To go deeper into the modeling of the real-world interactions, both of them have access to a dataset of real news (the KB). Indeed, both fakers and fact-checkers have access to the internet to do their respective task. The generator uses it to find images that can be repurposed and generate a fake caption that falsely describe the picture when the detector uses it to gather pieces of evidence to verify the integrity of a query package. The goal of the generator is to repurpose all available images to spread as much misinformation as possible while the discriminators' is to tell if a query package has been generated by its opponent or is real news.

Because the dataset could be really big (it is expected to contain possibly all validated knowledge about the world), the retrieval of related information needs to be optimized. In order to do so, multimedia packages are first encoded using modality-specific encoders. These embeddings are then stored and the retrieval is similar to a database query. Because finding nearest neighbors in such a big dataset is infeasible in practice, authors used efficient approximate search methods for similarity-based querying through the Faiss[8] library.

The training is done as follows: the generator first finds plausible misleading candidates, that are, K-most similar images with dissimilar caption. For example, it finds similar faces of different persons as shown in figure 11. The network then produces a caption by aggregating these candidates using a score for each of them, based on their similarity with the original metadata. The detector, on the other hand, first gathers pieces of evidence to validate the query package by retrieving K-most similar packages (using both the image and the metadata so there are two sets, one containing packages retrieved by image similarity, the other by metadata similarity). Feature vectors of these packages are then passed to a consistency verifier network that assesses the semantic integrity of the query package.

[8] https://github.com/facebookresearch/faiss



(a) Google Landmarks **(b)** IJBC-IRD **(c)** Painter by Numbers

Figure 11: Examples of similar image retrieved given the query one, [25]

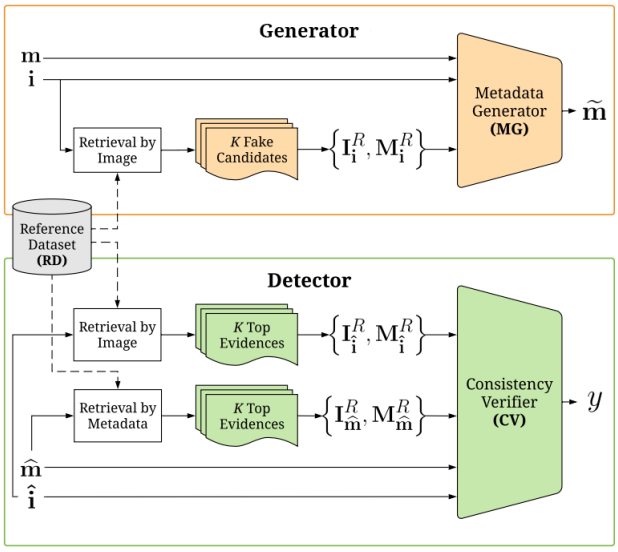


Figure 12: Illustration of the operation of the detector and the generator, [25]

Each component is trained to optimize their respective task: the detector is trained to optimize the probability of detecting an untampered package as such while the generator is trained to minimize the probability that the detector classifies its generated packages as tampered. These two modules' operation are illustrated in figure 12.

Authors did not test AIRD on previous datasets, arguing that even MEIR "falls short on the deceptive potential of entity-manipulations because it was created by randomly swapping the entity in a given caption with the same class of entity (person, organization, or location) from a caption in an unrelated package". Yet they used a similar approach on different datasets, such as Google Landmarks Dataset[9] for landscapes, IJBC-IRD[10] for faces and Painter by Numbers[11] for painting's

[9]https://www.kaggle.com/google/google-landmarks-dataset
[10]https://www.nist.gov/programs-projects/face-challenges
[11]https://www.kaggle.com/c/painter-by-numbers/data

artist by swapping entities but in even more similar packages than in MEIR, even if MEIR is not "randomly swapped" as they claim. They however tested the package comparison approach on these newly created datasets and it performed significantly worse than AIRD.

3.5 Evaluation

Evaluation metrics used to describe the performance of an image repurposing detection model are usuals metrics used to evaluate a classifier. That is, F1-measure to evaluate both the precision and the recall of the model and Area Under Curve (AUC) to evaluate every working point achievable given different classification thresholds. Two measures of F1 are used, tampered and untampered F1, respectively by considering tampered package as the positive class or considering untampered package as positives.

Results reported in table 1 show that the package comparison approach [23], while having similar performance than the semantic integrity one [20] on random manipulations, outperforms the last on the more challenging MEIR. However, in the case of the package comparison approach [23] and the adversarial learning [25] one, the comparison is harder to do. Even if the latter have better results on tested datasets as reported in table 2, it uses a different image encoder that is specialized for the kind of entity present in each dataset. This comparison is therefore not fair and does not allow to conclude on the superiority of an approach over the other.

Method	**MAIM**			**Flickr30K**			**MS COCO**			**MEIR**		
	F1 tampered	F1 clean	AUC	F1 tampered	F1 clean	AUC	F1 tampered	F1 clean	AUC	F1 tampered	F1 clean	AUC
Semantic integrity [20]	0.75	0.77	-	**0.89**	**0.88**	-	**0.94**	**0.94**	-	0.56	0.63	0.60
Package comparison [23]	**0.78**	**0.78**	**0.87**	0.88	**0.88**	0.95	0.92	0.92	**0.96**	**0.73**	**0.76**	**0.81**

Table 1: Performances comparison between semantic integrity approach [20] and package comparison one [23]

Methods	**Google Landmarks**			**IJBC-IRD**			**Painter by Numbers**		
	F1 tampered	F1 clean	AUC	F1 tampered	F1 clean	AUC	F1 tampered	F1 clean	AUC
Package comparison [23]	-	-	-	0.50	0.72	0.76	0.22	0.64	0.53
Adversarial learning [25]	**0.95**	**0.91**	**0.98**	**0.95**	**0.89**	**0.97**	**0.83**	**0.68**	**0.84**

Table 2: Performances comparison between adversarial learning approach [25] and package comparison one [23]

4 Contributions

While image repurposing is clearly defined as reusing an image out of context to convey false information, the task of image repurposing detection is still a bit ambiguous. From our perspective, previous approaches suffer from a lack of semantic content in texts used. Indeed, texts in AIRD's datasets are only the name of the person pictured, the artist of the painting or an identifier of the landscape and in the case of MEIR, text inputs are pictures' caption. The study of the correspondence of an image and an objective description is known as image-text matching and there is already a lot of publications on the subject [26, 27, 28, 29] as well as datasets for training and evaluation. We think that image repurposing detection differs from image-text matching in the way that the text modality is not a direct description of the image but a semantically richer corpus with information that goes beyond what is explicitly present in the image and is therefore more

challenging. Given that collecting examples of fake news is hard because nobody claims to write fakes news (except satirical newspapers that are not a great example of actual fake news) and can result in inconsistent format in corpuses that will be exploited by models, we think that generating convincing fake news is more viable and sustainable than collecting it. Created datasets then allow us to test the performance of a state-of-the-art multimodal representation on this task as well as exploring the possibilities to explain decisions of used models.

4.1 Datasets creation

Given the observation that datasets used in previous studies in the image repurposing field lacked of semantic content, the main objective during the creation of these new datasets was to create them in such a way that the model need to learn to extract the meaning of the image and the text instead of learning to recognize specific instances.

The use of professional press articles allows to tackle this problem because they are big corpuses often coupled with an illustration picture that gives a representation of their contents. Moreover, some presses have a "standard" in their articles, that is, some rules in the format and the writing style. This is helpful because the model won't find any biases in the format of the text and so will have to focus on the semantics themselves. This also allows the use of text generators that will write corpuses which follow these rules, resulting in a homogeneous dataset.

The news agency Reuters, in addition to include an image with nearly every articles and having a pattern in their format, provides an API by which every of their article is downloadable. A simple script can therefore iterate over their database to create a collection of real articles including headlines, snippets, full text articles, images and their objective textual descriptions. Part of this collection can then be perturbated in order to create a dataset composed of real and fake articles. To generate fake articles, there are two options: either tampering the text or the image.

4.1.1 Text generation

One way to create cases of image repurposing given the collection previously collected is to generate full text articles that will be associated with original images. One way to do so is to use a model able to generate corpus that can both follows the writting pattern of the news agency and be constrained to write on a given subject, defined for example by an headline. Modified versions of original headlines can then be given as input of such a model in order to create a fake article linked to the original image.

To keep the article close enough to the image so that the fake is not too obvious, the perturbation in the headline need to be subtile. The remplacement of an entity by another entity of the same type such as a person or a country by another one is a good exemple of a manipulation that can be misleading if the generated article gives information related to the new entity. A well known task in the natural language processing field is the named-entity recognition (NEM) which goal is to detect and identify such entities in a text. Models used to perfom this task are beyond the scope of this internship but there are various libraries that allow to perform state-of-the-art NEM such as NLTK[12] and SciPy[13]. These tools allow to identify entities in headlines and to swap them with entities of the same type. These modified headlines can then be used as inputs of the text generator to define subjects it will write on.

[12]https://www.nltk.org/
[13]https://www.scipy.org/

The standard in text generation is, for the moment, the model Generative Pre-trained Transformer-2 [15] (GPT-2), the second version of the GPT [30] model from OpenAI. This model is a stack of transformers decoders that have been pre-trained to generate corpuses. The Grover [31] model share the architecture of GPT-2 but has been trained on different newspapers datasets instead of diverse forms of text and is therefore able to generate corpuses that look like real articles from these newspapers. Moreover, it has been trained to generate an article conditioned on an headline and a "domain" (i.e. a newspaper website) so it is perfectly suited for creating fake news using modified headlines. An example of an article image along its original headline and text next to the perturbed version of the headline and the article generated by Grover can be found in figure 13.

Article image

Pakistan police clash with doctors protesting lack of virus protection gear

(Reuters) - Riot police wielding batons used force to break up a protest by Pakistani doctors and medical staff against a lack of gear to protect against coronavirus, arresting dozens of medics who say the government has failed to deliver promised supplies. Reuters journalists at the scene, in the southwestern city of Quetta, saw hundreds of doctors and paramedics, some in face masks and scrubs, chanting their demands. Some were dragged off by riot police in helmets, armed with rifles and batons. A senior police official said 30 protesters had been arrested for defying a ban on public gatherings imposed during a lockdown to fight the spread of the virus. Doctors threatened to stop working unless the detained protesters were released. Pakistan has reported a total of 3,277 cases of the virus, including 50 deaths. [...]

(A)

India police clash with doctors protesting lack of virus protection gear

(Reuters) - Indian doctors protested in front of Parliament on Friday in a rare show of unity against a government that they say is not sufficiently protecting them against the country's dreaded AIDS virus. India's healthcare system is struggling to cope with HIV/AIDS and is failing to achieve national goals for both treatment and prevention, one recent study showed. As the healthcare system struggles, doctors are fighting to boost testing of the BRCA1 and BRCA2 genes to find those carriers of the HIV virus who can take anti-retroviral drugs. In the first major test of health financing, about half the 13,000 doctors in Kolkata went on a strike in March this year to protest at not being given protective uniforms and rubber gloves during AIDS tests. [...]

(B)

Figure 13: Example of an original article (A) and one generated following our methodology with Grover (B), headlines are in bold

However, as will be discussed in section 5.1, the resulting dataset suffers from a bias due to the difference in subjects covered in the Grover training set and the news newly collected to generate our dataset. Fine-tuning Grover not being possible with the released code[14], we chose to fine-tune GPT-2 in order to quickly observe if this kind of bias could be corrected by training the generator with news from the same time period as real news in the dataset.

4.1.2 Image swapping

The other way to create cases of image repurposing is to exchange the original image of the article by another image. As seen in previous studies, there are different levels of difficulty achievable following this approach, depending on the way the exchange is done. Random swapping is indeed easier to detect than intelligent swapping.

The easiest way to trick a detection model is to replace an image by another that it finds visually similar or belonging to the same category (replace a face by another face, a building by another one, etc). Thereby, exchanging images using classification classes or the similarity of images representation extracted by a CNN will result in a more challenging dataset.

4.2 Detection models

While designing complex models can increase performance on a given task, it seems that even a very simple classification model can achieve really great results given a good representation of the input. Hence, pre-trained transformer encoders are gaining in interest because they can help to

[14] https://github.com/rowanz/grover

achieve good results on many tasks without the need to create complex architectures. Moreover, they lower the amount of data needed and can be used jointly with any other downstream models. Thus, their use seems like a "preliminary task" before even trying to build complex downstream models. Yet, even if BERT boosted their use in the NLP field, they are still seldom used in computer vision and in the multimodal field whereas their extracted representations seem to be much better than other methods' ones. The goal of this section is to study a state-of-the-art powerful multimodal representation and compare its performance on the image repurposing task against "simpler" models.

4.2.1 Baselines

Since previously described approaches are using a reference dataset to make the classification, the easiest way to empathize the usefulness of pre-trained transformers using our datasets is to use simpler models. However, it is necessary to study their improvements in the context of models described in section 3 in an upcoming study.

Our baseline models are really simple in order to limit the impact of downstream classification layers on results and make an accurate comparison of the quality of the extracted representation. First, text and image embeddings are extracted using pre-trained models (BERT for texts and ResNet50 for image) and condensed into lower dimension vectors using fully connected layers. These feature vectors are then passed through a fully connected softmax to make the prediction. Three different variations of this model have been implemented: one using only text input, one using only image input and one using multimodal representation through the concatenation of both modalities. These different architectures are illustrated in figure 14. The two monomodal models are expected to perform very poorly since one modality alone is not sufficient to detect image repurposing so they serve as a verification on the presence of a bias in the data which would make the task too simple.

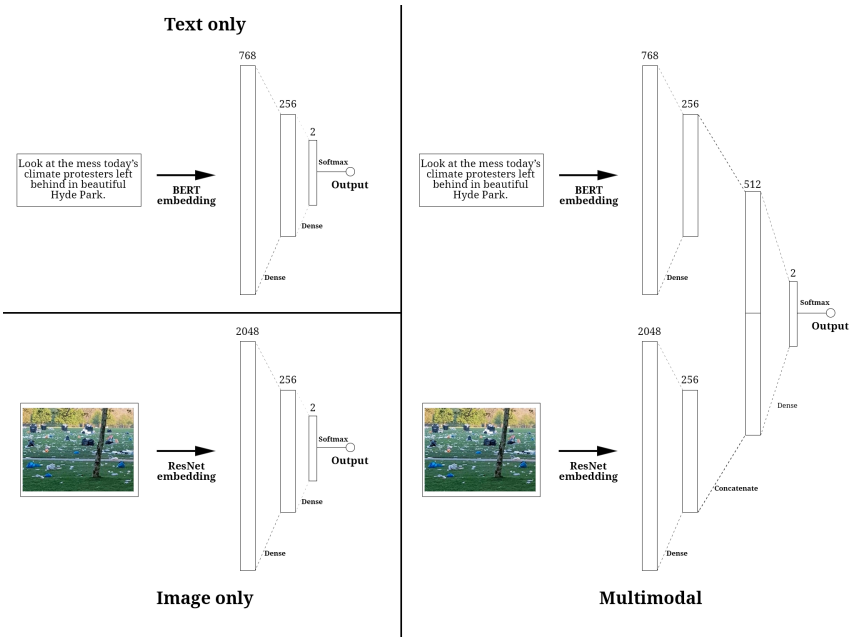


Figure 14: Architecture of the three baseline models

4.2.2 LXMERT

BERT's success in many natural language processing tasks has motivated the creation of large models that are pre-trained on very high level and abstact tasks and therefore can be applied with great results to various other tasks with little fine-tuning. Following this trend, many BERT-like models dealing with other input formats have been created such as image and video, but also multimodal inputs. These latter extend the attention mecanism used in BERT, besides having self-attention layers that allow a better understanding of relationships between parts of a single modality, they also have cross-attention layers. These layers allow to learn alignement and relationships between parts of the two modalities. While in the case of texts, the input is splitted in parts by tokenizing it (i.e. split a sentence in words or parts of words), images are divided into parts using Regional-Convolutional Neural Networks [32] (R-CNN).

R-CNN are an extension of CNN that first detect several regions (defined by bounding-box coordinates) containing objects in an image and then extract features from these independent areas instead of the whole image. An image can then be represented by a list of features describing different objects in it, which allow to align specific objects with specific words in the sentence and thereby to achieve a reasoning and matching of the two modalities as shown in figure 15.

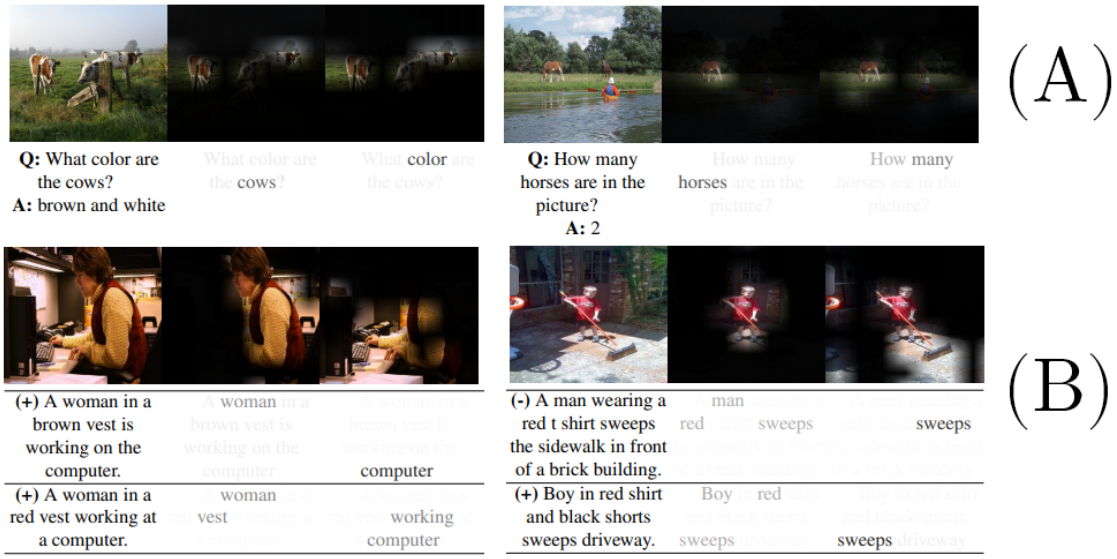


Figure 15: Examples of alignements done in matching (A) and reasoning (B), [33]

Some image-text BERT-like models have been proposed such as ViLBERT [34], VisualBERT [35], VL-BERT [36], UNITER [37] and LXMERT [38], yet they have a lot in common in their architecture and have similar results as shown in the table 4 of [37]. The implementation of LXMERT[15] and its R-CNN[16] being more accessible than others coupled with its ability to output both a representation of the package in a joint space and a representation of the text and the image in a common space led to the decision of using this model for the image repurposing detection task. A summary of following explanations of LXMERT's architecture can be found in figure 16.

[15] https://github.com/airsplay/lxmert
[16] https://github.com/airsplay/py-bottom-up-attention

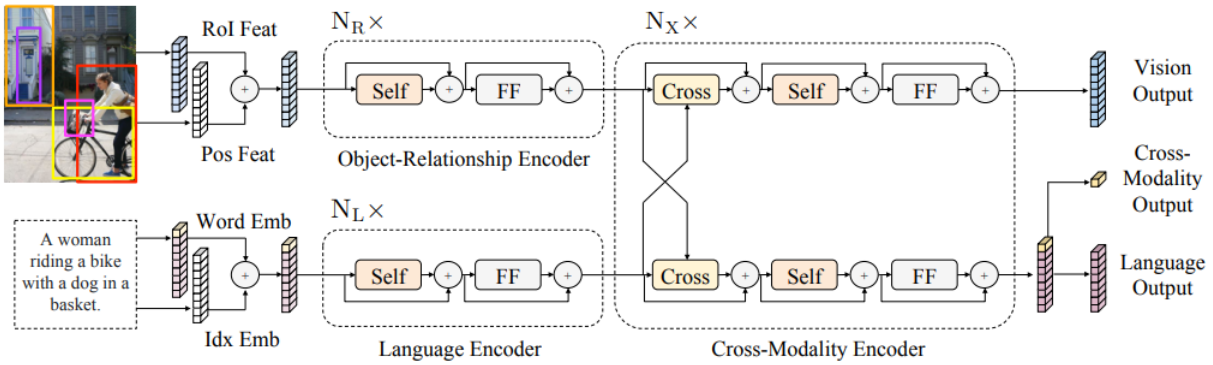


Figure 16: LXMERT architecture summary, [38]

Prior to the model itself, modalities are first splitted and embedded. For the text, the sentence s is split into n words ($s = \{w_1, \ldots, w_n\}$) using the same WordPiece tokenizer as BERT. Next, each word w_i and its absolute position in the sentence i are projected to vectors using embedding sub-layer. The two are then added together and normalized to produce the final embedding of the word.

$$\begin{aligned} \hat{w}_i &= \text{WordEmbed } (w_i) \\ \hat{u}_i &= \text{IdxEmbed } (i) \\ h_i &= \text{LayerNorm } (\hat{w}_i + \hat{u}_i) \end{aligned} \tag{5}$$

For the image, the R-CNN trained for image captioning of [39] is used. It firsts detects m objects in the image i ($i = \{o_1, \ldots, o_m\}$). Each object is then represented by its position features (i.e. its bouding box coordinates) and the features vector of this Region-Of-Interest (ROI). A position aware embedding is then learned by adding normalized outputs of two fully connected layers, thus providing spatial information that are useful for visual reasoning.

$$\begin{aligned} \hat{f}_j &= \text{LayerNorm } (W_\text{F} f_j + b_\text{F}) \\ \hat{p}_j &= \text{LayerNorm } (W_\text{P} p_j + b_\text{P}) \\ v_j &= \left(\hat{f}_j + \hat{p}_j\right)/2 \end{aligned} \tag{6}$$

After these embeddings layers, there is a serie of encoders: a **language encoder**, an **object-relationship encoder** and a **cross-modality encoder**. These encoders are built on the basis of two kinds of attention layers: *self-attention* layers and *cross-attention* layers. The operation of these layers has been detailed in the section 2.3.

The language encoder and the object-relationship encoder only focus on a single modality (language *or* vision). Each layer in a single-modality encoder contains a self-attention sub-layer followed by a feed-forward sub-layer that is composed of two fully-connected sub-layers. A residual connection is also added after each sub-layer. There is respectively N_l and N_r layers in the language encoder and the object-relationship encoder.

Outputs of these mono-modality encoders are then used as input of the cross-modality encoder. Each layer in the cross-modality encoder is composed of one bi-directionnal cross-attention sub-layer, two self-attention sub-layers and two feed-forward sub-layers. N_x cross-modality layers are stacked, i.e. the output of the k-th is used as the input of the $(k+1)$-th layer of the cross-modality encoder.

Inside a layer, the bi-directionnal cross-attention sub-layer is first applied, it contains two uni-directional cross-attention sub-layers: one from language to vision and one from vision to language. The query and context vectors are the output of the $(k-1)$-th layer. If we denote by h the language features and v the vision features:

$$\begin{aligned} \hat{h}_i^k &= \text{CrossAtt}_{\text{L}\to\text{R}}\left(h_i^{k-1}, \left\{v_1^{k-1}, \ldots, v_m^{k-1}\right\}\right) \\ \hat{v}_j^k &= \text{CrossAtt}_{\text{R}\to\text{L}}\left(v_j^{k-1}, \left\{h_1^{k-1}, \ldots, h_n^{k-1}\right\}\right) \end{aligned} \tag{7}$$

These cross-attention sub-layers are used to exchange information and align entities between the two modalities in order to learn joint cross-modality representations. Deeper internal connections are then built using self-attention sub-layers applied to the output of the cross-attention sub-layer:

$$\begin{aligned} \tilde{h}_i^k &= \text{SelfAtt}_{\text{L}\to\text{L}}\left(\hat{h}_i^k, \left\{\hat{h}_1^k, \ldots, \hat{h}_n^k\right\}\right) \\ \tilde{v}_j^k &= \text{SelfAtt}_{\text{R}\to\text{R}}\left(\hat{v}_j^k, \left\{\hat{v}_1^k, \ldots, \hat{v}_m^k\right\}\right) \end{aligned} \tag{8}$$

The k-th layer outputs are then produced by feed-forward sub-layers on top of these self-attention outputs followed by a residual connection after each sub-layer.

LXMERT model has three outputs, for language, vision and cross-modality respectively. The language and vision outputs are the feature sequences generated by the cross-modality encoder while the cross-modality output, following the practice in BERT, is obtained by appending a special token [CLS] before the sentence and using the corresponding features vector as the cross-modality output.

In the same way as BERT, LXMERT has been built to serve as basis to solve various tasks following the *pretrain-then-transfer* learning approach. This means that the model is pre-trained on very large datasets in order to learn a good representation of the input space and can then be fine-tuned easily on a lot of different tasks. Indeed, the output of LXMERT can be used as the input of a downstream model, freeing from the need of a complex architecture and a big dataset to achieve good results thanks to the expressive representations that it extracts. The addition of LXMERT can also be done by plugging it to a downstream model and finetune both at the same time. This means that LXMERT can be plugged to nearly every existing model.

As a first experiment to use LXMERT on the image repurposing task, we chose to follow the architecture used by authors for Visual Question Answering [40] which is similar to our baselines. In this configuration, dense layers are fed with the multimodal output to make a classification as illustrated in figure 17 and LXMERT is fine-tuned along with classification layers. Since in our baselines, BERT and ResNet are not fine-tuned during the training, we also mesured performance of this architecture without fine-tuning LXMERT to perform a fair comparison.

4.3 Explainability

Deep learning models, despite their efficiency, are difficult to use in practice. Indeed, they are called *black-box* models because their inner operations are not understandable by humans, mostly because of their depth, thus it is very complicated, if not impossible, to explain why a model made a particular decision. In addition to ethical issues raised by this observation, there are cases where explanations are not only desirable but needed. In the case of fake news, pure censorship done by a machine is inconceivable, so the main goal in the image repurposing detection is not only to flag a fake news as such, but also to express what looks suspicious in order to help humans to double check the classification and take action if needed.

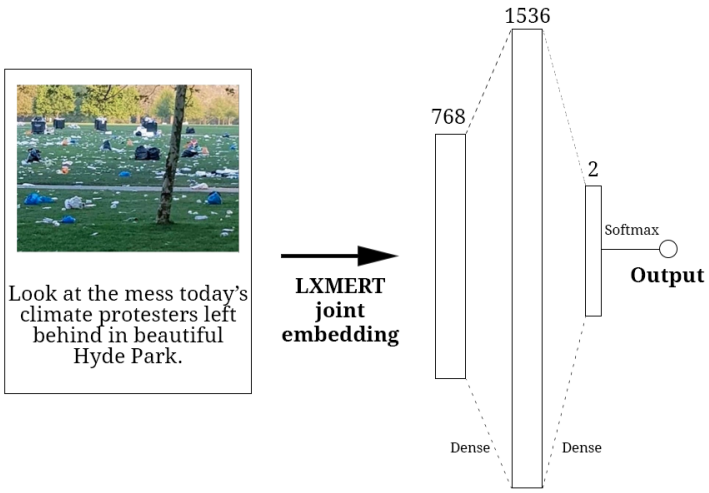


Figure 17: Architecture of the classification model using LXMERT

Moreover, has shown in [41] and [42], a model can in reality learn a systematic bias present in datasets used during the training and the evaluation. While the example in the latter is the presence of snow in the background of wolves pictures and grass in dogs ones, systematic biases can come in various form when dealing with news. Yet, the intended behavior of the model is to study semantics of the image and the text, not to study the presence or absence of an unique word or symbols. Such biased models will perform poorly in real-life situation where this particular bias will not be present. So explanation of the classification, in addition to allow the acceptance and use of the model, allows to discover and correct a bias in the dataset to train truly effective models.

Explanations can be either global or local, respectively if they explain the behavior of the model in general for every possible input or for a given instance. In the case of fake news detection, it is better to get local explanations that give precise evidences to help moderators validate or invalidate a classification quickly and easily. The two following approaches have been tested on our models to correct biases in datasets and check if explanations they provide would suit for a real life usage.

4.3.1 LIME

Authors of [42] introduced Local Interpretable Model-agnostic Explanations (LIME). This method allows to extract *local* explanations of a classification, that is, what part of the input played a role in the decision. It is said *model-agnostic* because it can, in theory, be applied to any type of classifier, no matter its input form or complexity.

To give an explanation, LIME first splits the input into *interpretable elements*, i.e. parts of the input that can play a role in the classification such as words for texts or super pixels (group of similar pixels) for images. Then, various inputs similar to the one being explained are created by abblating some of these elements. The study of the variation in the classification of these inputs allow to highlight which part of the original input is important for the classification. Indeed, if the classification score is lower without a word, it means that the word played *positively* in the classification while if it is higher then it means that the word played negatively. This process allows to attribute score to each word that represent their role in the classification as shown in figure 18.

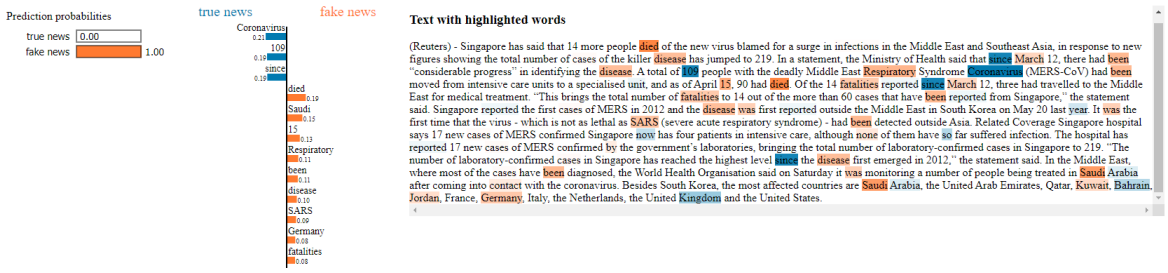


Figure 18: Visualisation of a LIME explanation, blue words played for the "true news" classification while orange ones played for "false news"

While in theory, this method can be applied to any type of input and model just by defining:

1. The form of an interpretable element for the given input
2. The method to extract these elements from the input
3. The method to ablat these elements in such a way that the classification is affected by their absences and not the way they were ablated itself (for example, make every pixels of the super pixel white can be seen as snow and not only the absence of element)

in practice, the explanation can be inaccurate. This is the case when dynamic embeddings are used, because the absence of a word also changes the embedding of others so the deletion of the word affects the classification more than by its absence alone, the third point can therefore not be satisfied. Moreover, when the task needs a complex analysis of the text's semantic, words individually are not enough to explain the decision. Finally, even if it can be applied to images or text, there is still little to no application of LIME to multimodal inputs. For all of those reasons, LIME is not suited for real life utilisation of proposed models.

4.3.2 Attention maps

As explained in section 2.3 attention allows, in addition to improve results on various task, to better understand the functional behavior of a model. Indeed, the observation of attentions maps calculated during the processing gives an overview of the model reasoning that can be used as local explanations. The study of BERT's inner operations gained in interest in order to explain its very good results and to see what information it encodes. Tools like exBERT [43] have even been build to make the exploration of these attention maps easier.

While most studies like [44] are focused on patterns that show the overall functioning of BERT such as structural and syntactic relations as shown in figure 19, attention maps can also highligh a bias in the data. Most observable patterns are related to language rules and are common to every model and dataset so if an attention map does not follow these "general rules", studying it might give information that explain the model's decision.

For example, if the attention of every word is focused on a single -non special or punctuation- token, it indicates that the word was important in the decision. If this pattern shows everytime the word is present in a sentence, it can indicate that this word has a prominent place in the decision and this might be due to a bias in the data such as the omnipresence of this word in a class.

Unlike in the case of LIME where the analysis is limited to the impact of a word, attention maps can highligh very complex behavior both in a single attention map but also across different attention layers and heads. However, searching an explanation through all the attention maps is tedious and the creation of an automatic extraction of explanations is needed to be used in real life scenarios.

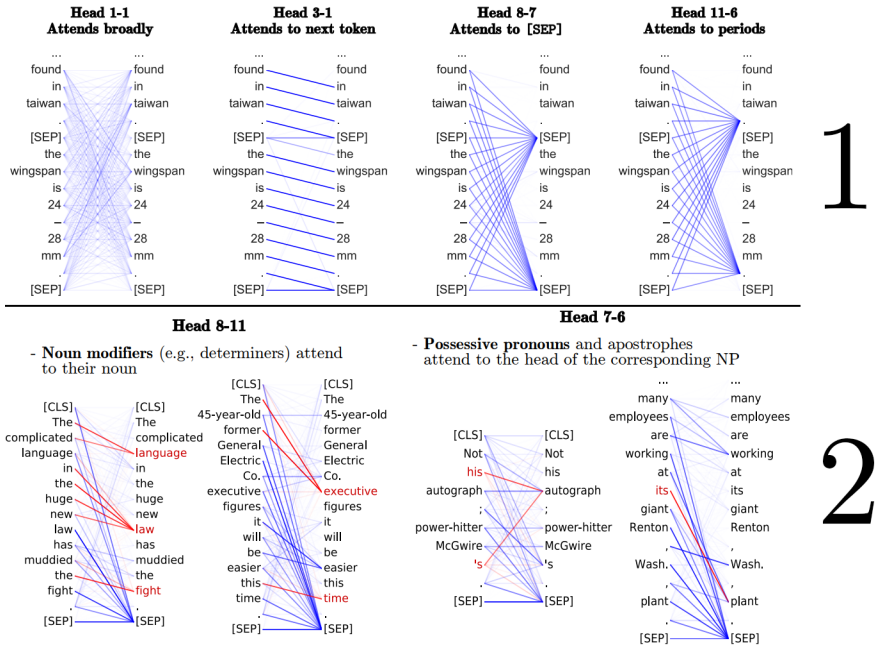


Figure 19: Visualisation of BERT's structural (1) and syntactic (2) patterns, [44]

5 Results

Now that the creation of the datasets, the models used as well as the explanability approaches have been described, this section will focus on information that experiments allowed us to obtain. They concern both the difficulties in creating unbiased datasets and the performance of LXMERT on the image repurposing task.

5.1 Biases in generation

Text generation Feeding Grover with perturbed headlines and the domain "reuters.com" resulted in very convincing news. Yet, as shown in table 3, the model that only has text as input achieved 75% of accuracy, highlighting a bias in the dataset since a text-only approach on an unbiased dataset cannot be better than a random classification, that is to say 50%. This means that generated news were too different from the scrapped news. Biases in generated data are most of the time in the form of generation artefacts, that is, mistakes in the generation that make the result less organic. However, there was not such artefacts in Grover's fakes news and there was nothing observable on a specific article which could be discriminating. The recurrence of the importance of certain words in the explanations of LIME and the attention maps led to a global

study of tokens' frequencies in each label that emphasized a more subtile bias. As shown in table 4, there is a clear difference in subjects covered in generated and scrapped news; while real news focus more on the actual COVID-19 pandemic (coronaviru, pandem, lockdown, covid, nineteen, crisis), fake news focus on other coronaviruses epidemics (middl, east, syndrom, respiratory, mer (for Middle East Respiratory Syndrome, MERS), saudi, arabia, sar (for Severe Acute Respiratory Syndrome, SARS)). Even if the word "coronavirus" appears as much in true and fake headlines, its omnipresence (nearly 70%) in the real news results in a bias. Indeed, since Grover has been trained *before* the 2020 coronavirus epidemic, it only "knows" about previous coronaviruses and not "the coronavirus" (COVID-19) itself.

While this bias is exacerbated by the very important coverage of the actual crisis, this illustrates the dynamism of the information, which is an important aspect to considere when dealing with fake news generation but also detection. A detection model should be able to learn new informations to stay up to date and be able to process correctly new subjects that appear on a daily basis. The same implication holds for the generation model if such model is used to train the detector.

	Grover dataset	GPT-2 dataset
Multimodal baseline	74%	**87%**
Text-only baseline	75%	**87%**
Image-only baseline	50%	50%
LXMERT fine-tuned	**88%**	79.8%
LXMERT	68%	65%

Table 3: Accuracy of baseline and LXMERT models on datasets based on text generation

However, the additional data needed to fine-tune the generator increases the total amount of data needed while the available amount of data on this period remain constant. This forces a compromise between sizes of datasets for the generator and for the detector. As a consequence, using fine-tuned GPT-2 corrected the frequencies bias but the generated data contains artefacts due to the lack of training data, resulting in another bias which distort results of models as reported in table 3.

These observations show that it is necessary to first pre-train a model to generate proper articles using an huge collection of old articles and perpetually fine-tune it on recent news so it is able to generate texts about new subjects. This perpetual training is needed since the detector also need to be trained continuously on newly generated data to keep being up to date and make good prediction on new topics. This dynamic aspect in training is pretty unusual since most deep learning models training and fine-tuning are based on big one time process rather than regular small updates and is a major challenge to create fake news detection models.

Image swapping Even with the implementation of a similarity search using Faiss' indexes that allow efficient approximate neighbors search in vector space such as ResNet embeddings, the creation of a dataset by swapping image with a similar one is still not trivial. Indeed, while when randomly swapping images there is no problem in using only images from news in the dataset because there won't be any duplicate in a permutation, it is possible that two or more images share the same "closer" image in the dataset and this resulted in a bias since some images was present multiple times and was even both in the training and the test set. This is illustrated by the high result of the image-only model on the similarity dataset in table 5.

token	frequency difference	frequency in scrapped news	frequency in Grover's news
coronaviru	42,94%	**69,54%**	26,60%
percent	30,97%	2,04%	**33,01%**
nineteen	22,23%	**30,23%**	8,00%
pandem	21,04%	**25,81%**	4,76%
edit	20,93%	0,23%	**21,16%**
middl	19,29%	3,18%	**22,46%**
east	19,00%	4,14%	**23,14%**
respiratori	17,87%	3,86%	**21,72%**
saudi	16,79%	4,59%	**21,38%**
arabia	16,34%	4,25%	**20,59%**
covid	14,97%	**15,31%**	0,34%
syndrom	14,75%	0,40%	**15,14%**
sar	14,41%	1,02%	**15,43%**
crisi	14,29%	**24,90%**	10,61%
day	13,90%	**37,15%**	23,26%
mer	13,78%	0,11%	**13,90%**
report	13,22%	40,61%	**53,83%**
ad	13,05%	**30,86%**	17,81%
twenti	13,05%	**56,04%**	42,99%
lockdown	12,88%	**14,35%**	1,47%
organ	12,37%	11,46%	**23,82%**
home	12,14%	**23,82%**	11,68%
year	11,40%	50,82%	**62,22%**
deadli	11,29%	1,82%	**13,10%**
kill	10,89%	11,74%	**22,63%**

Table 4: Top tokens' frequencies differences between scrapped news and Grover's one

Removing the image from the index after it has been used once and swapping images only in the same set would lead to "default swapping" where there won't be any close image in the dataset, thus being similar to random swapping. For these reasons, it is necessary to build an auxilary image set with enough images to prevent default swapping while not having duplicates in the dataset.

	Random - Article	Random - Title	Random - Image description	Similarity - Article
Multimodal baseline	**53%**	49%	48%	51%
Text-only baseline	**53%**	49%	49%	51%
Image-only baseline	51%	51%	51%	60%
LXMERT fine-tuned	50%	55%	**85%**	**72%**
LXMERT	50%	**56%**	78%	58%

Table 5: Accuracy of baseline and LXMERT models on datasets based on image swapping

5.2 LXMERT performances

It is hard and risky to conclude on biased results, yet it seems that in the case of the presence of a bias in a modality, LXMERT does not achieve good results without fine-tuning. This might be due to the fact that, by default, LXMERT extracts a balanced representation of image and text and therefore cannot exploit this bias by focusing only on the biased modality while fine-tuning allows the extracted representation to be more focused on the biased modality. In the case of our baselines, however, since both representations are separated, the exploitation of a biased modality is possible. This is observable in the table 3 where results of the multimodal and the text-only baselines are the same, meaning that the image provide no additional information and that only the text is used.

Three variations of the abstraction in the text of the random swapping dataset have been studied in order to explore the behavior of LXMERT: one using full text articles (Random - Article), one using titles (Random - Title) and one using image descriptions (Random - Image description). In the first and the second variation, the relation between the image and the text is not explicit and, as a result, LXMERT is poorly performing compared to the last variation (which is very close to the image-text matching) where both are directly linked as reported in table 5. This can be explained by the operation of cross-attention. Since it tries to align parts of the image with parts of the text, it performs very poorly when there is no (explicit) alignment possible. So, LXMERT multimodal representations seem to not be suited in the case of image repurposing. However, further tests need to be performed to totally discard cross-attention usage in image repurposing detection:

1. Feed cross-attention layers with abstracted representations to see if it can align "concepts" and therefore empathize a more abstract link between the image and the text

2. Use the two separate monomodal outputs of LXMERT that are representations of the image and the text in a common space to see if it performs better than other representations used in previous approaches

6 Conclusion

After describing state-of-the-art of image repurposing and modalities embeddings, we studied the possibility to generate a training dataset for this task, the performance of a model that performed really well in others multimodal tasks and tools to explain its decisions.

While it is hard to collect fake news to build a dataset, generating them is also an arduous task due to the many types of bias that can be introduced during the generation. However, given the fact that fake news detection models need to be trained regularly to stay up to date and that collected data can also suffer from biases, building a unbiased generation pipeline would result in a perennial source of training data for these dynamic models.

Pre-trained transformers using cross-attention, although very efficient on tasks where the text and the image are directly linked, seem to be less effective when they are not. Nevertheless, tests were limited to the multimodal joint representation of LXMERT and using its distinct common representation of text and image could lead to better results.

Finally, it is crucial to be able to provide explanations on the classification because it is useful both to monitor the presence of a bias and to help humans validate the decision since this kind of model should never be used to censor information without human supervision.

During the remaining time of the internship, experiments will focus on generating proper datasets by correcting biases through a better pre-training of GPT-2 and the creation of an auxiliary dataset of images on one hand and testing the possibility offered by the two monomodal output of LXMERT on the other hand.

References

[1] Regina Marchi. With Facebook, Blogs, and Fake News, Teens Reject Journalistic "Objectivity". *Journal of Communication Inquiry*, 36(3):246–262, 2012.

[2] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

[3] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *International Conference On Machine Learning, ICML*, 2009.

[4] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations, ICLR*, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition, CVPR*, 2016.

[6] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *Neural Information Processing Systems, NIPS*, 1993.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Neural Information Processing Systems, NIPS*, pages 3111–3119, 2013.

[8] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *CoRR*, abs/1411.2539, 2014.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521:436–44, 2015.

[10] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1:318–362, 1986.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Neural Information Processing Systems, NIPS*, 2017.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics, NAACL-HLT*, 2019.

[14] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations, ICLR*, 2018.

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. `https://openai.com/blog/better-language-models`, 2019.

[16] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *International Conference on Computer Vision, ICCV*, 2015.

[17] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal Deep Learning. In *International Conference On Machine Learning, ICML*, 2011.

[18] Vedran Vukotić, Christian Raymond, and Guillaume Gravier. Bidirectional Joint Representation Learning with Symmetrical Deep Neural Networks for Multimodal and Crossmodal Applications. In *International Conference on Multimedia Retrieval, ICMR*, 2016.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Neural Information Processing Systems, NIPS*, 2014.

[20] Ayush Jaiswal, Ekraam Sabir, Wael Abd-Almageed, and Premkumar Natarajan. Multimedia Semantic Integrity Assessment Using Joint Embedding Of Images And Text. In *Association for Computing Machinery Multimedia, ACM-MM*, 2017.

[21] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support Vector Method for Novelty Detection. In *Neural Information Processing Systems, NIPS*, 1999.

[22] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *International Conference on Data Mining, ICDM*, 2008.

[23] Ekraam Sabir, Wael AbdAlmageed, Yue Wu, and Prem Natarajan. Deep Multimodal Image-Repurposing Detection. In *Association for Computing Machinery Multimedia, ACM-MM*, 2018.

[24] Michael Goebel, Arjuna Flenner, Lakshmanan Nataraj, and B. S. Manjunath. Deep Learning Methods for Event Verification and Image Repurposing Detection. *CoRR*, abs/1902.04038, 2019.

[25] Ayush Jaiswal, Yue Wu, Wael AbdAlmageed, Iacopo Masi, and Premkumar Natarajan. AIRD: Adversarial Learning Framework for Image Repurposing Detection. In *Computer Vision and Pattern Recognition, CVPR*, 2019.

[26] Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Visual Semantic Reasoning for Image-Text Matching. In *International Conference on Computer Vision, ICCV*, 2019.

[27] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked Cross Attention for Image-Text Matching. In *European Conference on Computer Vision, ECCV*, 2018.

[28] Tianlang Chen, Jiajun Deng, and Jiebo Luo. Adaptive Offline Quintuplet Loss for Image-Text Matching. *CoRR*, abs/2003.03669, 2020.

[29] Nicola Messina, Fabrizio Falchi, Andrea Esuli, and Giuseppe Amato. Transformer Reasoning Network for Image-Text Matching and Retrieval. *CoRR*, abs/2004.09144, 2020.

[30] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. `https://openai.com/blog/language-unsupervised`, 2018.

[31] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending Against Neural Fake News. In *Neural Information Processing Systems, NIPS*, 2019.

[32] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition, CVPR*, 2014.

[33] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. Dual Attention Networks for Multimodal Reasoning and Matching. In *Computer Vision and Pattern Recognition, CVPR*, 2017.

[34] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *Neural Information Processing Systems, NIPS*, 2019.

[35] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. *CoRR*, abs/1908.03557, 2019.

[36] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-BERT: Pre-training of Generic Visual-Linguistic Representations. In *International Conference on Learning Representations, ICLR*, 2020.

[37] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: Learning UNiversal Image-TExt Representations. *CoRR*, abs/1909.11740, 2019.

[38] Hao Tan and Mohit Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*. Association for Computational Linguistics, 2019.

[39] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In *Computer Vision and Pattern Recognition, CVPR*, 2018.

[40] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision, ICCV*, 2015.

[41] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5):1–42, 2019.

[42] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *North American Chapter of the Association for Computational Linguistics, NAACL-HLT*, 2016.

[43] Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformers Models. *CoRR*, abs/1910.05276, 2019.

[44] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What Does BERT Look At? An Analysis of BERT's Attention. *CoRR*, abs/1906.04341, 2019.