4. Model

4.1 Bag-of-words (BOW) and frequency–inverse document frequency (TFIDF)

One of the simplest NLP methods applied for our baseline is a bag-of-word or count vectorization. The vocabulary of size V, number of unique words, can be considered as a vector of length V where each dimension corresponds to a word. Hence, each unique word is one-hot encoded. If this framework is applied for a whole sample phrase, the text fragment can be represented by a sparse vector with V dimensions, each dimension corresponding to a unique word. Each dimension gives the number of occurrences of the corresponding word in the sentence. This tool might highlight potential terms that may make incoherences more identifiable.

The second method is TF-IDF (term frequency and inverse document frequency). The term frequency corresponds to raw count $f_{t,d}$ of a term t in a document d divided by the total number of terms in d:

$$\text{tf}(t, d) = f_{t,d}$$

The inverse document frequency is the log of the total number of documents N divided by the number of documents in set D in which the term t appears:

$$\text{idf}(t, D) = \log(\frac{N}{|d \in D : t \in d|})$$

The tf-idf of term t in document d from the set of documents D is given by:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

By doing so, a term that occurs in many documents will have a tf-idf score closer to 0 while a term that appears in a restricted subset of the corpus will get a higher score. This may help

us to partially filter out the noise from most frequent terms. When applied to our problem, the weight of each word included in a sample is penalized if it occurs in too many observations.

In both cases, we use a maximum of 4096 features corresponding to words in the vocabulary ordered by term frequency.

4.2 Word embeddings and language modeling

In order to distinguish incoherent from coherent texts in our next experiments, we rely on words embeddings obtained from the skip-gram model described in (Mikolov et al., 2013a). In NLP, a vector representation can be obtained from words by training a neural network on the text corpus. This neural network has one hidden layer with a size h (number of hidden units) that determines the dimension of the word vectors. The input layer is a vector that has the size n of the vocabulary (set of unique words/tokens available in the corpus) and is presented as a one-hot vector with 1's corresponding to the position of a single item in the vocabulary. The output layer has the same size as the input layer and its activation function is a softmax that enables multi-classification.

In the skip-gram model in particular, the goal of the softmax is to provide probabilities for all words of the vocabulary that a word is a neighbor of the input center word within a predefined window range (by default, the window is five-word long). With this approach, we may predict the most likely output neighboring word (it corresponds to the highest real value found in the output vector from the softmax classifier) in the context surrounding the input word (represented by the input one-hot vector). The probability of a context output word given an input center word is formulated by:

$$y_j = p(w_o|w_c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

where w_o and w_c are respectively the output context word and the input center word, u_o and v_c are respectively the word representations (vectors) for output word and center word.

Formally, over the entire text corpus, the model has to maximize an objective function ((Rong, 2014), (Goldberg and Levy, 2014)), that is the probability of any context word given a certain input word:

$$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m<j<m \\ j\neq 0}} p(w_{t+j}|w_t;\theta)$$

where t is an index that spans every word of text length T, j is the index of the output context word w_{t+j} located within the window of size m around the input center word w_t and θ represents the parameters to be optimized being the weights of the hidden layer. The objective function may also be simplified by taking the log probability:

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m<j<m \\ j\neq 0}} \log p(w_{t+j}|w_t;\theta)$$

With the `gensim` library from (Řehůřek and Sojka, 2010), we train a skip-gram model on the French most recent version of Wikipedia articles. All Wikipedia editor-oriented syntax (<ref> references <ref>, [hyperlinks]) are removed from the text training set. Once the training is done, we extract the weights W of the hidden layer that will serve as word embeddings. They correspond to a matrix with n rows (number of unique words of the vocabulary, in our case it amounts to 729,651 with the French Wikipedia) and h dimensions (real values, they usually represent 100 or 300 features). The words that co-occur together within the same window range will have features that set them close to each other in the vector space.

Vector representations have the capacity to capture syntactic and semantic properties from the training corpus, as explained in (Mikolov et al., 2013b). For instance, one of the most famous examples shows the existence of a "feminine vector" that connects man to $woman$ or $uncle$ to $aunt$ in the language vector space or of a "CEO-firm" vector that connects $Steve\ Jobs$ to $Apple$ and $Bill\ Gates$ to $Microsoft$. Another illustration comes from the vector operation $king - man + woman$ that leads to a position near the word $queen$. These examples illustrate the existence of vectors defined by the following pattern: $word1$ is to $word2$ what $word3$ is to $word4$. Our intention is to take advantage of these embeddings properties for classifying coherent and incoherent text as in (Taddy, 2015).

In the mean vector experiment, we use the word vectors of all elements in a sample string and compute the mean vector from them. Each dimension of that mean vector corresponds to a feature that is used for our models. We alternatively use a 100-dimensional and a 300-

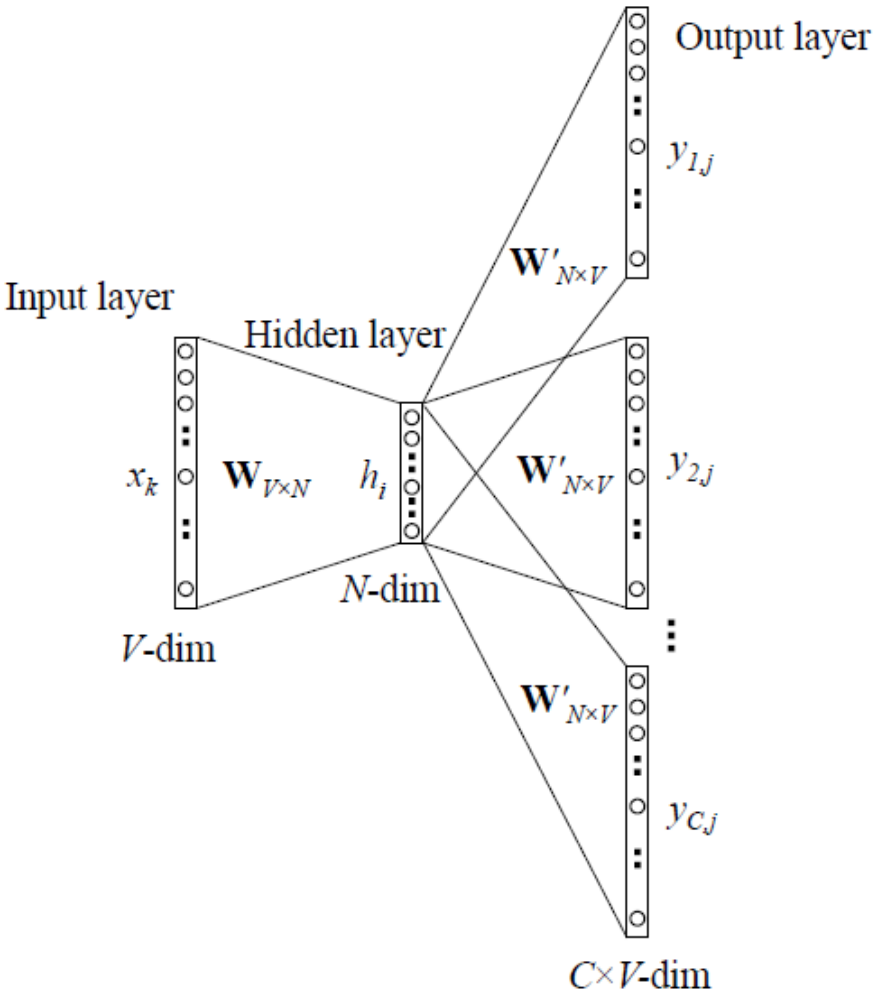

Figure 4.1: Illustration of a Skip-Gram model from (Rong, 2014)

dimensional language model that will make 100 or 300 features for each sample. Increasing the number of dimensions of a language model allows corresponding word vectors to better capture semantic and syntactic properties. Beyond 300 dimensions, the benefits are too marginal and computationally expensive.

In the skip-gram model experiment, we use the tools provided in the skig-gram model of the `gensim` library for computing the likelihood of a whole sentence. It spans each element of the text sample and take into account each of its closest neighboring words. At each of those steps, the program computes the log probability of each (center input word, context output word) pair and eventually sums of of them for getting the log probability of the entire text fragment. The log probabilities of all samples are used as unique feature for our predictions. We also alternatively use a 100-dimensional and a 300-dimensional language model.

For all four experiments (BOW, TFIDF, mean vector and skip-gram model), we use logistic regression and decision tree classifiers. We attempted to apply random forests but after progressively reducing the number of tree estimators, they appeared to be too complex models relatively to the dataset.

The logistic regression model embodies the sigmoid function that is relevant for a binary classification task:

$$\sigma(y) = \frac{e^y}{e^{-y} + 1} = \frac{1}{1 + e^{-y}}$$

with $y = \beta_0 + \sum_{j=1}^{K} \beta_j x_j$ where β_j is the coefficient for each feature and x_j are the parameters to be optimized with gradient descent.

In the case of the decision tree classifier, the splits and the thresholds are defined so as to maximize the drop in impurity that can be defined as either classification error, entropy or gini impurity.