Jordi Pascual Fontanilles

Analysis of the communication of smart city features through social media

MASTER'S THESIS

Supervised by Dr. Antonio Moreno Ribas

Computer Security Engineering and Artificial Intelligence Master's Degree

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2020

Abstract

Thanks to the evolution of the internet to the Web 2.0 and the fact that we live in a knowledge society, a vast amount of information is generated nowadays. Several disciplines have interest on analysing all this data. Particularly, from the communication field, there is a lot of interest on analysing how tourist brands communicate, and about what topics they are communicating. Smart cities are entities which through social networks such as Twitter, communicate with residents and tourists about the different dimensions that compose a smart city. In this document, a new method for the automated analysis of the communication of Destination Management Organizations and institutions of smart cities on their Twitter accounts is proposed. The proposed method, makes use of semantic comparisons of words by using vector representations of words. The techniques explored to obtain the vector representations of words are word embeddings. This new method has also been used to analyse 9 different smart cities Twitter accounts for 3 different smart cities dimensions, sustainability, economy and technology. The results obtained by the system are compared with smart cities rankings in the beforementioned smart cities dimensions, showing that there is no correlation between the amount of communication for a smart city dimension, and having a good position in the ranking of that dimension.

Agraïments

En primer lloc, voldria agrair a l'Antonio Moreno i l'Aïda Valls per haver-me donat l'oportunitat de realitzar aquest treball, el qual he gaudit realitzant-lo. També per la seva ajuda, pel seu temps i dedicació, els coneixements que m'han transmès i els seus consells, orientació i suport.

També a tots els meus amics, que en tot moment m'han animat, i m'han donat tot el seu suport sempre que ho he necessitat.

Finalment, he d'agrair sobretot a la meva família, per la seva ajuda i suport incondicional, sense el qual no hagués pogut realitzar aquest projecte ni aquests estudis.

Table of contents

List of tables

List of figures

1. Introduction

It is undeniable that we live in a knowledge society. In it, knowledge that may be used to improve the human condition is generated, shared and is available to everyone conforming the society. All this knowledge is in fact raw data, which has been increasing with the growth of Information and Communication Technologies (ICT). And not just the amount of data that is generated has increased, also the speed at which it is generated.

Moreover, with the evolution of the internet to the Web 2.0, people have been given tools that allow them to connect with each other, to use the information available on the internet, and most importantly, to produce new information and share it on the internet. This is why the Web 2.0 is also known as participative and social web, because it focuses on the content generated by people and in its participation on the web.

The vast amount of knowledge that is generated in the Web 2.0 has raised a lot of interest from several disciplines to analyse all the available information. Particularly, from the communication field, there is a lot of interest on analysing how tourist brands communicate, and how this communication is perceived by people. This is possible thanks to the information generated by tourists, visitants and Destination Management Organizations (DMO), which are institutions that make use of social media to communicate with residents and tourists.

In previous works, such as (Lalicic, Huertas, Moreno, & Jabreel, 2020), (Jabreel, Moreno, & Huertas, 2017), (Lalicic, Huertas, Moreno, & Jabreel, 2019) or (Jabreel, Huertas, & Moreno, 2018), the emotional brand communication of diverse DMOs were analysed and a new framework to automatically perform this analysis at large-scale was proposed. This framework is based on the semantic comparison between the tweets adjectives and concepts which can be found on WordNet. In this work, a new semantic comparison method based on vector representations of words is wanted to be tested. The techniques that are used to obtain these vector representations from words are called word embeddings.

Smart cities are another topic which has been widely analysed. They do not have a uniform and widely accepted definition. They are complex entities that through the use of technology, and especially ICT (Kummitha & Crutzen, 2017), aim to efficiently manage resources and services. For instance, achieving resource saving, sustainable development, improvements on the residents' quality of life or supporting the decision making process of all the city stakeholders. This is possible by using the insights gained from data of the city that has been retrieved and analysed.

The technologies and ICTs adopted by the smart cities provide both tourists and residents easy access to the information and services they dispose (Encalada, Boavida-Portugal, Ferreira, & Rocha, 2017). This fact fomented the appearance of Smart Tourist Destinations (STD), which aim to support the different dimensions of smart cities. These multiple dimensions of smart cities can be analysed from several perspectives. Some important aspects of smart cities are sustainability issues (e.g. how the city promotes the use of

renewable energy sources or electric vehicles), technological issues (e.g. which kind of technologies are used in the city to capture data and analyse it) or economic issues (e.g. how the city promotes innovation, entrepreneurship or the appearance of technological start-ups).

In a previous work, a framework named SA was proposed to analyse and rank STDs in some of their different dimensions. (Huertas, Moreno, & Tran, 2019) (Roig, Moreno, & Tran, 2018) (Tran, Huertas, & Moreno, 2017) To obtain the data needed to perform the analysis and the ranking, surveys and interviews were performed to the DMOs of the analysed STDs, as well as data from secondary data sources, such as governmental reports on tourism policies and regional statistics or information of official tourism websites.

Because DMOs and institutions make use of social media to communicate with residents and tourists, it should be possible to analyse the content that DMOs of STDs share on their social media. This should allow knowing which of the multiple dimensions of smart cities are communicated for each of the STDs.

The social media Twitter is one of the most used social media by users, tourists and DMOs. This is because of its ease of use, the great frequency of publications that is allowed, the links to other sites and its interaction in real time. Analysing the data found on the Twitter accounts of the STDs, the beforementioned analysis of their communication could be performed in an automated way.

1.1. Objectives

The main goal of this work is to develop a methodology that can automatically analyse a set of tweets retrieved from a Twitter account of a Smart City and detect if they are related to concepts of smart cities. This main goal is divided in the following sub-objectives:

- Search for the official Twitter accounts from cities appearing in a smart city ranking, and selection of those which are relevant for the analysis.
- Retrieval of the tweets sent by the selected accounts.
- Definition of new multi-level ontologies that cover the main concepts to be considered in smart cities for some of their multiple dimensions.
- Analysis of the different word embeddings on the literature and their properties.
- Selection of the word embedding that is more suitable for this work.
- Use of word embeddings to analyse the retrieved tweets and automatically detect if they are related to the concepts of the build ontologies.
- Analyse the obtained results and compare them with the rankings of smart city.

1.2. Document structure

The rest of this document is organized as follows:

- Chapter 2: a brief state of the art on diverse methods used to analyse tweets is presented. First, methods that have been used to analyse tweets for a tourism specific

case. They use semantic comparisons between adjectives on the tweets and concepts that can be found on WordNet. Second, other methods that make use of word embedding techniques jointly with other techniques.

- Chapter 3: diverse natural language processing tools are explained. They are divided in two main sections. First, techniques used to preprocess sentences and words. Second, diverse word embedding techniques, which is the tool that is intended to be used in this work for language modelling and feature learning.
- Chapter 4: the process followed to build ontologies related with smart cities aspects is described. For this work three ontologies have been built. They are for the sustainability, economy and technology smart cities aspects.
- Chapter 5: the methodology used for the proposed system is described. Each of the three parts that compose the system, the preprocessing of the tweets, the preprocessing of the ontologies and the classification of the tweets are explained. They use the different techniques explained on chapter 3 and the ontologies built on chapter 4. Moreover, examples of the process are shown.
- Chapter 6: the experimental results of the proposed system are shown. The smart cities Twitter accounts to be analysed are selected. Different word embeddings are tested using the system in order to select one of them. It is used to analyse the selected smart cities Twitter accounts.
- Chapter 7: a list of conclusions and some lines of future work are presented. The conclusions include the process followed to develop the proposed system, as well as the conclusions from the analysis of the results obtained from the system. The future work proposals focus on expanding the analysis done by expanding and creating more ontologies, and on trying to overcome some of the limitations of the proposed system.

2. State of the art

In this chapter there is a brief overview of the state-of-the art studies that are related to this work. The first ones describe methods that have been used to analyse tweets for a tourism specific case. The second ones are other works that analyse tweets using word embeddings jointly with other techniques in order to classify them for different purposes.

Word embeddings are one of the most used Natural Language Processing techniques to model language and learn features. They are able to represent words in a vector representation such that words which have a similar context are represented in vectors that are close in the vector space.

Thanks to the properties of the vector representations that are learnt after training a word embeddings model, they could be used to analyse the contents of tweets sent by DMO.

2.1. Previous work on tweet analysis on tourism

The following works try to analyse the emotional brand communication of diverse DMOs. In (Lalicic, Huertas, Moreno, & Jabreel, 2020) they used data from Facebook and Twitter aiming to know which emotional values are best for DMOs to engage with their users on social media for the top 10 most popular DMOs in Europe according to the data provided by TripAdvisor in 2017. An extended version of this work was also published with an extended analysis on the obtained results. (Lalicic, Huertas, Moreno, & Jabreel, 2019)

In (Jabreel, Moreno, & Huertas, 2017) they analysed 10 destinations selected from the top 25 most popular destinations in Europe according to TripAdvisor. In this case, they just used data from Twitter, analysing a total of 60000 tweets. The objective of the work was to compare the communication of the emotional values from the destinations Twitter accounts with the communication by the tourists in their personal Twitter accounts. A similar work made the same comparison for 9 main European tourist destinations, analysing a total of 54000 tweets from the DMOs of the destinations and their residents. (Jabreel, Huertas, & Moreno, 2018)

In all these works the method used for the analysis of the tweets is the same. It is based on the semantic analysis between the adjectives found on tweets and the emotional values they defined. The emotional values they defined are 5, sincerity, excitement, competence, sophistication and ruggedness, which have a total of 54 associated subcategories. These 54 subcategories are the words which are compared with the adjectives found on the tweets.

The similarity measure that was used is the Wu and Palmer semantic similarity measure. It requires a structured knowledge base, and they decided to use WordNet, which is a lexical database of semantic relations between words. The equation used to compute the similarity between two words $c1$ and c2 is the following one.

$$Sim_{W\&P}(c1, c2) = \frac{2*depth(c3)}{length(c1,c3)+length(c2,c3)+2*depth(c3)} \quad (1)$$

In it, $c3$ is the Least Common Subsumer of $c1$ and $c2$ in WordNet. Length is a function that returns the number of hypernym links among two concepts, being a hypernym a generic term

whose semantic field includes other terms. Finally, depth is a function that returns the number of hyperlinks that separate a word from the root of WordNet.

The similarity value can range from 0 to 1, and a threshold has to be set to determine which concepts are considered similar enough. The similarity value compared with the threshold is the maximum one obtained from the similarity measures between the synsets associated to each of the words in WordNet. These synsets are synonymous words which express the same concept. Following, a table shows some similarities examples computed using their proposed method.

Word 1	Word 2	Wu & Palmer similarity
beauty	glamour	0.93
beauty	charm	0.87
beauty	honesty	0.66
beauty	original	0.63
beauty	innovation	0.55

Table 1: Wu & Palmer semantic similarity examples

2.2. Other work on tweet analysis

Some proposed approaches use the vector representations of the words of the tweet directly as an input of a classifier. For instance, in (Dai, Bikdash, & Meyer, 2017), they wanted to classify the tweets according to a topic of public health such as influenza. They use the vector representations of the words of tweets to perform a clustering of the words using an adaptation of the Chinese Restaurant Process of Dirichlet Process. The similarity measure to determine if a word is clustered into an existing cluster or to a new cluster is the cosine similarity between the word vectors. Finally, each cluster vectors are aggregated using the average operation, and the resulting aggregated vectors are compared with the topic vector using the cosine similarity. If the similarity is higher than a determined threshold, the tweet is classified to that topic.

A similar approach was proposed by (Dabiri & Heaslip, 2018). They determine if a tweet is classified or not into a category by computing what they call the tweet Similarity Index (TSI). Previous to the classification of the tweets, a set of words related to the categories in which tweets have to be classified are selected. In their tests they gathered words related to traffic, as they wanted to classify tweets between traffic or not traffic. The word embeddings of those sets of words are then averaged. The resulting vector is the representation of a category. Then, for a labelled training set, TSI, which is the average of the cosine similarities between the words of a tweet and the topic vector, is computed. The resulting TSI for both categories is averaged, resulting in the threshold used for the classification of tweets.

More complex approaches were also proposed. For instance, (Villena-Román, Cobos, & Cristóbal, 2014) proposed a method for tweet classification in real time focused on mining the opinions of citizens to obtain their trends, opinions, etc. The categories into which the tweets are classified are defined in ontologies. To classify the tweets in the diverse categories, a hybrid approach involving two classifiers is used. The first classifier uses supervised machine learning techniques for multi-label classification. In order to train and use the classifier, the tweets are tokenized into words, and words are mapped to lower dimensional vectors. Then, they use K-Nearest Neighbour, even though other supervised learning classifiers such as Naïve Bayes, Support Vector Machines, etc, are also suitable to perform the classification. The second classifier used is rule-based, and its objective is to fine tune the results obtained from the machine learning classifier. Rules are defined per category and are based on logical expressions of natural language terms that accept or reject the classification to the categories.

Another framework proposed by (Alkhammash, Jussila, Lytras, & Visvizi, 2019) uses Ontotext (Ontotext, 2016) to semantically enrich the tweets. The words from the tweet are annotated in different entities, and a confidence score is provided for each of them. There is a total of 8 different entities, as for instance person, location or organization. Some heuristics are used to classify the words into the entities by using a labelled dataset.

An approach introduced in (Dabiri & Heaslip, 2018) determines if a tweet is from a certain category or not by using a neural network. This neural network is built on top of the word embeddings, and a manually labelled tweets dataset is used to train it. The types of neural networks they tested are Convolutional Neural Networks (CNN), Long Short-Term Memory neural networks (LSTM) and a combination of both kinds of networks. The best results they obtained when classifying tweets into traffic related and non-traffic related are using word2vec word embeddings and a Convolutional Neural Network classifier.

The proposal of (Li, Shah, Liu, Nourbakhsh, & Fang, 2016) uses a modification of word embeddings called topic enhanced word embeddings (TEWE). This version of word embeddings allows having different word vectors for each word depending on the topic. By using TEWE one of the main problems of word embeddings is solved, which is that they have just one vector for word, hence, they cannot represent polysemy. Jointly with TEWE, there is an entity knowledge base (entity KB). This knowledge base is built from data specialized on social media, providing more data that can enrich the topical context of a tweet. A sequential minimal optimization (SMO) classifier is trained using both the entity KB and TEWE to classify the tweets.

Another hybrid proposal by (Farajidavar, Kolozali, & Barnaghi, 2017) fuses the results of two word embeddings to classify city-related events. The first word embedding is trained to extract the syntactic information of the words by using a convolutional neural network. The second one is trained to extract the semantic information by using a Conditional Random

Field. A Restricted Boltzman Machine is finally trained to fuse the output labels of both embeddings. The models are trained to classify tweets consisting on just one sentence.

The proposal of (Pereira, Pasquali, Saleiro, & Rossetti, 2017) combines horizontally the matrices obtained from the word embeddings bag-of-words and bag-of-embeddings. By using the resulting combined matrix, tweets can be binary classified. The classifier techniques they tested are LinearSVM, Logistic Regression and Random Forests. The best results were obtained by the LinearSVM.

Finally, the authors of (Fand, Macdonald, Ounis, & Habel, 2016) have defined new metrics to evaluate the coherence of the topics in tweets which are based in word embeddings. The new metrics are compared with Pointwise Mutual Information (PMI) and Latent Semantic Analysis (LSA) metrics. The topic modelling methods used to create the topics are the Latent Dirichlet Allocation, Twitter Latent Dirichlet Allocation and Pachinko Allocation Model, and are compared pairwise. The obtained results show that the metrics using word embeddings are able to capture the coherence of topics on tweets and are more robust and efficient than metrics based on PMI and LSA.

2.3. Summary

In this chapter the state of the art methods related to this work have been summarized. Their objective is to analyse tweets, and they do so by using different approaches.

The first described methods analyse the emotional brand communications of DMOs, and they do so by using the Wu and Palmer semantic similarity measure. The second described methods use word embeddings to obtain the representations of the words that allow to classify the tweets.

In the following section different Natural Language Processing tools that will be used for the proposed system are explained. Different word embeddings are also described in that section.

3. NLP Tools

In this section different tools used in natural language processing that are used for the system that we propose are explained. First, methods that are used in the preprocessing of sentences and words. Second, word embeddings, that is the tool used for language modelling and feature learning.

3.1. Preprocessing tools

The following tools are used to preprocess sentences, with the exception of the last one which is used to preprocess words. The preprocessing tools are needed in order to obtain the important information of the input text in the correct way so it can be processed afterwards.

3.1.1. Tokenizer

The tokenizer is used to extract the different words that form a sentence. In order to obtain these words or tokens, the characters that separate the different words are found. Multiple characters are used to find the separation of words, such as blank spaces, a point, a comma, etc. (BIrd, Klein, & Loper, 2009)

For instance, the tokenization of the tweet “We're a very proud city as well #LoveOurCitizens” would result in the following list of tokens [We, re, a, very, proud, city, as, well, #LoveOurCitizens]. In this case the separators used are the blank space and the apostrophe.

3.1.2. Stop words

Stop words are commonly used words which do not add any significant meaning to a sentence. They are removed from the sentence before the processing of it to reduce its computing time. There is not a single list of stop words. The following table shows some words considered as stop words for this work.

a	about	above	after	again	against	all	am	an
and	any	are	as	at	be	because	been	before
being	below	between	both	but	by	can	did	do
does	doing	don	down	during	each	few	for	from
further	had	has	have	having	he	her	here	hers
herself	him	himself	his	how	i	if	in	into
is	it	its	itself	just	me	more	most	my

Table 2: Words considered as stop words

Following, an example of the removal of stop words. The tweet “Aberdeen Maritime Museum is closed to visitors until further notice due to the Fire Alarm system failing.” would have removed the stop words which are coloured.

3.1.3. Part-Of-Speech Tagger

Part-Of-Speech tagging (POS-tagging) consists on classifying the words on a sentence into their part-of-speech. That is, classifying each word into the lexical category it has. Obtaining this information is needed for other methods, such as n-gram chunking, that will be explained in the following section.

A simplified table containing the lexical categories can be seen following. (BIrd, Klein, & Loper, 2009) There is a more extensive list of lexical categories which is the one that will be used in the real system. Here, a simplified list is shown with the most important categories.

Tag	Meaning	Examples
ADJ	Adjective	new, good, high, special, big, local
ADV	Adverb	really, already, still, early, now
NOUN	Noun	year, home, costs, time, Africa
NUM	Numeral	twenty-four, fourth, 1991, 14:24
VERB	Verb	is, say, told, given, playing, would

Table 3: Part-Of-Speech simplified lexical categories

Finally, the simplified lexical categories of an example tweet "The new solar panel system at Harry Sotnik house will save around 34 tonnes of carbon each year." can be seen following.

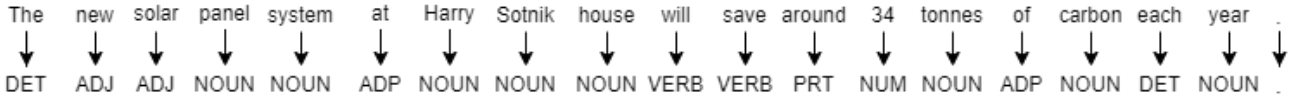

Figure 1: Part-Of-Speech Tagging example

3.1.4. n-gram chunking

In a text there exist idiomatic phrases and the ordering of words is meaningful. In order to get the ability of representing n-grams in word embeddings, some methods are needed. In word embeddings, n-grams are considered as single tokens even though they are formed by multiple words. Part of the preprocessing of the sentences must be finding and joining the n-grams as if they were a single word, so they are represented just as in the word embedding.

An approach is using chunking. The first step of chunking is to tokenize a sentence and obtain its POS-tagging as it has been explained in the sections before. Then, a grammar is used to find words that form an n-gram. The grammar is represented as a regular expression using the tags that can be assigned by the POS-tagger.

The regular expression used to detect the n-grams is *<DET>?<ADJ>*<NOUN>*. The determiner is optional, whereas there must be at least one adjective and a noun. This grammar is able to find noun phrases (NP) on sentences. Those NPs are n-grams, and if they consist on multiple words after removing the stop words, they are joined in a single token.

In the figure next, the NPs detected by the grammar are shown.

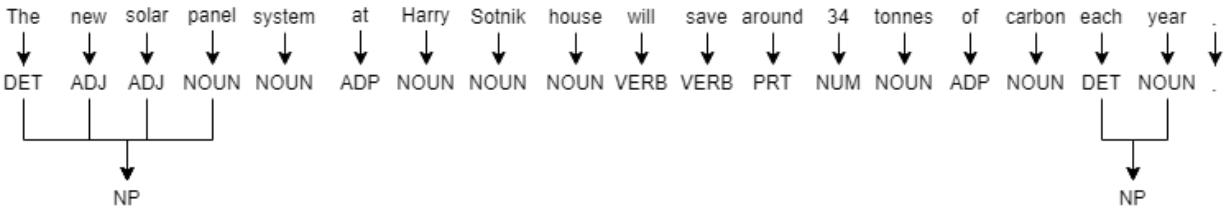

Figure 2: n-gram chunking example

The NPs found are "The new solar panel" and "each year". After removing the stop words, the only NP still having multiple words is the first one, which would be "solar panel". This n-gram would be joined and considered as a single token or word.

3.1.5. n-gram collocation model

Another approach to obtain n-grams on sentences proposed by (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) to learn the vector representation for phrases is to introduce the commonly found together words as new tokens on the vocabulary. Even though they called them phrases, those commonly found together words are effectively n-grams. As an example, the n-gram "New York" would be introduced as a new single token, whereas "this is" would not. The objective is the same as in n-gram chunking, finding n-grams and join them in a single token, as they are in the word embeddings.

They wanted to form a reasonable number of phrases without increasing the number of tokens on the vocabulary too much, to maintain the computational cost. The approach is data driven. For a pair of tokens, a score is computed and if it is higher than a threshold parameter, the tokens are considered a phrase and are joined in a new token. The score is computed with the following equation

$$score(w_i, w_j) = \frac{count(w_i w_j) - \delta}{count(w_i) \cdot count(w_j)} \quad (2)$$

where δ is a discounting coefficient that avoids joining infrequent words, $count(w_i w_j)$ the number of times both words appear together, and $count(w_i)$, $count(w_j)$ the number of times w_i and w_j appear respectively.

To allow joining phrases composed by more than two words, the computation can be repeated several times decreasing the δ coefficient, allowing the creation of longer phrases.

Another approach proposed by (Bouma, 2009) relies on the NPMI (Normalized Pointwise Mutual Information). PMI measures the probability of two events co-occurring in respect of what it would be expected taking as a basis the probabilities of both events considered independent. For this application, the events are two words being next to each other in a text.

It can be computed as

$$i(w_i, w_j) = ln \frac{p(w_i, w_j)}{p(w_i) \cdot p(w_j)} \quad (3)$$

were $p(w_i, w_j)$ is the probability of two words appearing together and $p(w_i)$, $p(w_j)$ are the probabilities of w_i and w_j appearing on the text respectively.

To obtain the normalized PMI, NPMI, they applied the following normalization to normalize the lower and upper bound of the equation.

$$i_n(w_i, w_j) = \left(ln \frac{p(w_i, w_j)}{p(w_i) \cdot p(w_j)} \right) \Big/ - \ln p(w_i, w_j) \quad (4)$$

With the normalization, the words can be scored to define if they have to be joined as a phrase or not. When $i_n(w_i, w_j)$ is 1 it means that both words only appear together in the text, and when is –1, that they never appear together. As in the first approach, two words are joined in a new token if their score is higher than a defined threshold.

Some examples of n-grams or phrases that would be detected by the two approaches explained are the following ones.

Artificial intelligence
Solar panels
Black Friday
Olympic torch

Table 4: Example n-grams found by n-gram collocation model

3.1.6. Word segmentation

In text, especially in social networks, there are multiple words that are joined in a single string without introducing blank spaces between the words. A clear example are hashtags in social networks.

The objective of word segmentation is to find the boundaries of the different words composing a single string without blank spaces, so they can be extracted. It is not a trivial task. For a string with n characters, there are 2^{n-1} possible word segmentations.

The approach proposed by (Norvig, 2009) solves this problem by dividing the string in different valid candidates and choosing the candidate with a higher probability in a defined probability model.

The splitting of the words has a word length delimiter to reduce the combinations. They propose to use a length delimiter of 20 letters per word. The way to split it, is in a first word and the remaining text, and apply recursively the splitting process to the remaining text.

The probabilistic model returns a probability for each word independently. For words that appear in the vocabulary, it is computed as $count(word)/\,N$, where N is the size of the vocabulary. For words that do not appear in the vocabulary, the proposal is to use $1/N$ as the default probability. Moreover, a probability that depends on the length of the unseen word can also be used, as a word with 6 letters is more probable than a word with 20 letters.

The word segmentation that is chosen is the one that gives a higher probability in the multiplication of the probability of the first word and the probability of the remaining ones.

Some examples of word segmentations applied to hashtags can be seen next.

Hashtag	**Word segmentation**
#carbonneutral	carbon neutral
#sustainable	sustainable
#smartcities	smart cities
#climateemergency	climate emergency

Table 5: Word segmentation example

3.2. Word embeddings

Word embeddings are a set of methods that aim to learn the representation of words in a vector space. Each of the words is mapped to a unique vector of real numbers.

There are multiple approaches to learn the word representations. For instance, using co-occurrence matrices (Lebret & Collobert, 2013) (Levy & Goldberg, Neural word embedding as implicit matrix factorization., 2014), probabilistic models (Globerson, Chechik, Pereira, & Tishby, 2007), explicit representations (Levy & Goldberg, Linguistic regularities in sparse and explicit word representations., 2014) or neural networks.

The objective is that words with similar context, have vectors which are close in the vector space. To compute the similarity in the vector space, cosine similarity can be used. It is computed as the dot product between the two vectors divided by the multiplication of both vectors normalized using l2-norm. The result can range from –1 to 1, being 1 that both vectors are close in the vector space, and –1 that are not close in the vector space.

In the following subsections, different word embedding models based on neural networks are explained.

3.2.1. Word2Vec

The word2vec word embedding can produce the vector representations using either of the two following architectures.

Continuous Bag-of-Words Model (CBOW)

The task for the CBOW model is to predict a word given its surrounding words. This task is created in order to use supervised learning techniques, but the important part of the resulting model is not the output, they are the weights of its hidden layer. The weights that are obtained after training the model on a corpus, are the word embeddings.

The corpus contains a vocabulary of words, which is of size V. The architecture of the neural network is simple, it has an input layer, a hidden layer and an output layer, as can be seen in the following figure.

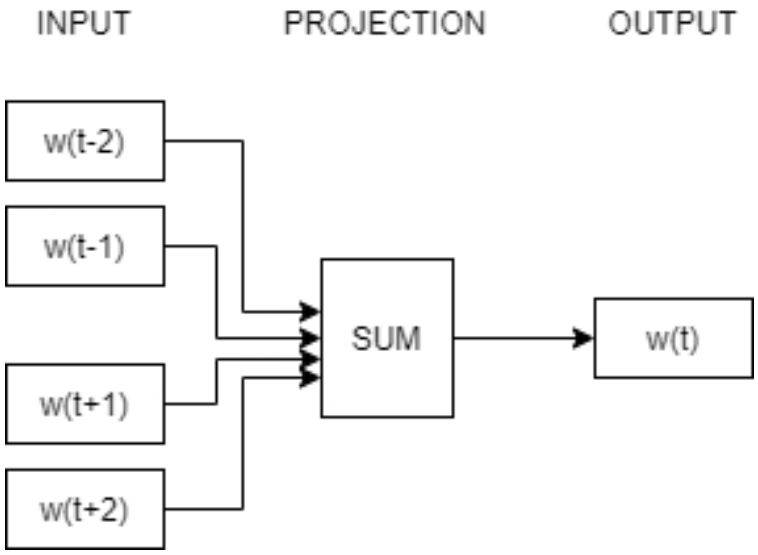

Figure 3: CBOW architecture

In the input layer, the n surrounding words of a word w_t in the training corpus are introduced. There are future and history words, $[w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n}]$. The best results that can be obtained is using four history and four future words. (Mikolov, Chen, Corrado, & Dean, 2013) Before introducing the words on the training data to the neural network, they are converted to one-hot encoding vectors of size V.

The hidden layer has D neurons, which defines the dimensionality of the word embeddings. Each of the V sized input vectors are then multiplied with the $V \cdot D$ hidden layer, resulting in as many vectors as input words. Each of those E size vectors are then averaged, obtaining the final activation which is fed to the output layer.

The output layer uses the softmax function to obtain the output of the neural network. It is a vector with the size of the vocabulary, V, which contains the probabilities of each word to be w_t.

Finally, a back propagation is performed to update the neural network weights according to the correct wt.

After the training is completed, the $V \cdot D$ hidden layer matrix weights represent the word embeddings. Because all the input words are projected to the same matrix, the order of the words does not influence this matrix, which can be seen as a projection.

Continuous Skip-gram

This neural network model is similar to the CBOW one. The main difference is the fake task it has to accomplish. In this case, from a word w_t, it has to predict its n surrounding words $[w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n}]$. The value of n is a random value in the range $[1, C]$, where C is a defined parameter.

The architecture is also the same, with an input, hidden and output layer.

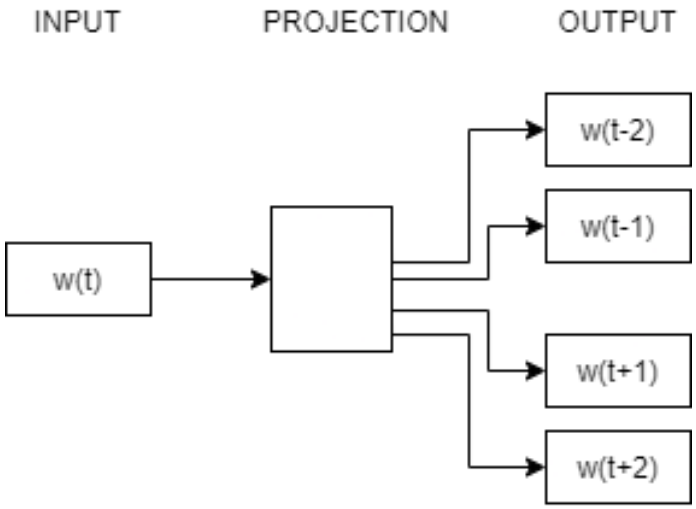

Figure 4: Skip-gram architecture

In this case, there is just one input vector which is also one-hot encoded into a vector of size of the vocabulary, V. The hidden layer also has D neurons which will correspond to the dimensionality of the resulting vectors. The input vector is multiplied by the hidden layer with size $V \cdot D$. The resulting vector of size D is then fed into the output layer.

The output layer uses a softmax classifier to obtain the probabilities for each of the surrounding words. That is, $2n$ vectors of size V are obtained. Then, their errors are computed, and the sum of the error vectors is back propagated.

After the training is completed, the $V \cdot D$ hidden layer matrix weights represent the word embeddings.

3.2.2. Word2Vec improvements

In this subsection, some methods that are commonly used to improve the word2vec neural network models are explained.

Hierarchical softmax

The softmax function used in the Skip-Gram model to get $p(w_O|w_I)$ is defined as

$$p(w_O|w_I) = \frac{\exp(v'_{w_O} \cdot v^T_{w_I})}{\sum_{w=1}^{W} \exp(v'_w \cdot v^T_{w_1})} \quad (5)$$

where v_w is the input vector representation of w, v'_w is the output vector representation of w, and W is the size of the vocabulary.

Because this computation has a computing cost of $\nabla \log p(w_O, w_I)$ which grows proportionally to the size of the vocabulary *W*, the softmax function can be replaced with hierarchical softmax for a more efficient computation. (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013)

Hierarchical softmax just needs to evaluate $\log_2 W$ nodes instead of *W*. This is possible because the output layer of the neural network is represented using a binary tree, as for instance a Huffman binary tree.

The hierarchical softmax function is

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w, j))]\!] \cdot v'_{n(w,j)} \cdot v^T_{w_I}) \quad (6)$$

where $n(w, j)$ is the j-th node on the path from the root of the binary tree to *w*, $ch(n)$ an arbitrary fixed child of *n*, and $[\![x]\!]$ is 1 if x is true, and –1 otherwise. In this case, the computing cost of the hierarchical softmax is proportional to $L(w)$, which in average is lower than $\log(w)$.

Negative sampling

An alternative to hierarchical softmax is the Noise Contrastive Estimation (NCE). It is intended to differentiate the target output from a noise sample by using a logistic regression classifier. It can maximize the log probability of the softmax, but because the purpose of Skip-Gram is to learn vector representations of the words, NCE can be simplified while maintaining the quality in the vector representations. The simplification of NCE is called Negative Sampling (NEG). (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013)

The NEG objective is

$$\log \sigma(v'_{w_O} \; v^T_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-v'_{w_i} \, v^T_{w_I}) \right] \quad (7)$$

which replaces every $\log P(w_O \mid w_I)$ the Skip-Gram objective. $P_n(w)$ is a noise distribution and is a parameter that must be defined. The unigram distribution $U(w)$ raised to the 3/4rd power, as for instance $U(w)^{\frac{3}{4}} / Z$, is the noise distribution which gives better results. (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013)

Subsampling of frequent words

In the training data there is a big imbalance between frequent and rare words. The co-occurrences of words like “Paris” and “France” together provide a better vector representation of the words than “France” and “the”. To balance this, and hence, getting more valuable information, subsampling is used.

To apply subsampling to a training set, each word in it is discarded with a probability $P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$, where *f* is a function that returns the frequency of a word, and t is a threshold that has to be chosen. With a threshold of around 10^{-5}, the results of training a Skip-Gram model are not only improved, but also take less time to train. (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013)

3.2.3. GloVe

The GloVe embeddings are a hybrid approach between window-based models such as word2vec and count-based models, such as capturing the frequency of co-occurrences in the training data. (Pennington, Socher, & Manning, 2014)

This hybrid approach allows GloVe to make a more efficient use of statistics that allow a faster training of the models while capturing rare words on the training data.

For a pair of words w_i and w_j that might co-occur, GloVe tries to minimize the difference between the inner product of both word embeddings and the logarithm of their number of co-occurrences. To accomplish this, the following objective function was proposed.

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \widetilde{w}_j + b_i + \widetilde{b_j} - \log X_{ij}\right)^2 \qquad (8)$$

In it, w_i and b_i are respectively the word vector and bias of the word i, $\widetilde{w_j}$ and $\widetilde{b_j}$ are respectively the word vector and bias of the word j, X_{ij} the number of times the word i occurs in the context of the word j, and finally, f is a weighting function that assigns a low weight to rare and frequent co-occurrences. Multiple functions could be used as f, but the one which has given them best results is

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & \begin{matrix} if\ x<x_{max} \\ otherwise \end{matrix} \\ 1 \end{cases} \qquad (9)$$

and the values that performed better on their tests are $x_{max} = 100$ and $\alpha = \frac{3}{4}$.

Because the co-occurrence counts can be encoded as a co-occurrence matrix, GloVe takes this matrix as its input instead of the whole training corpus as in the other word embeddings.

3.2.4. fasttext

In fasttext embeddings, the main difference with respect to skip-gram is that each word is considered as multiple character n-grams, and the word embedding of a word is the addition of the embeddings of each character. (Joulin, Grave, Bojanowski, & Mikolov, 2016) For

instance, the word “artificial” with $n = 3$ would be represented in fasttext as [“ar”, “art”, “rti”, “tif”, “ifi”, “fic”, “ici”, “ial”, “al”].

After representing the words as character n-grams, a skip-gram model like the one in the figure next is trained to learn the vector representations of the character n-grams. Once the character n-grams vector representations have been learnt, they can be used to obtain the vector representations of words.

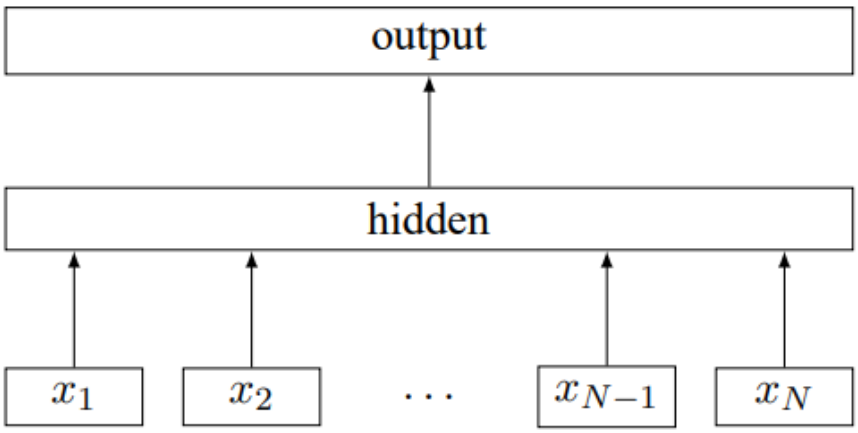

Figure 5: fasttext architecture

This approach allows generating better embeddings for rare words, and to handle out-of-vocabulary words. Because words are split into character n-grams, words not on the training corpus can be broken down into character n-grams and get its vector representation.

3.2.5. Results in vector space

After training the word embeddings, they all learn the vector representations for the words. Moreover, and the most interesting thing, is that using the vector representations, semantic and syntactic questions can be solved by performing algebraic operations on the vector representations. (Mikolov, Chen, Corrado, & Dean, 2013)

For instance, performing the computation $X = vector(\text{"}biggest\text{"})$ - $vector(\text{"}big\text{"}) + vector(\text{"}small\text{"})$ and searching the most similar word to X using the cosine similarity, the word “*smallest*” is obtained.

One example of a semantic task that can be performed using the learned vector representations is getting a man-woman relation. For instance, finding the word *W* in *brother* is at *sister* the same as *grandson* is at *W*. The correctly provided answer by the word vectors is that *W* equals *granddaughter*.

Another example of a syntactic question is finding opposite words. For instance, finding the word *W* in *possibly* is at *impossibly* what *ethical* is at *W*. Again, the vector representations can solve this task, giving the answer as *W* equals *unethical*.

Following, a table summarizing different tasks that can be solved using the vector representations learned can be seen. The task that is solved by the embedding is to find the

second word of the Word Pair 2 given the relation between the two words of the Word Pair 1. (Mikolov, Chen, Corrado, & Dean, 2013)

Type of relationship	**Word Pair 1**		**Word Pair 2**	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Table 6: Tasks solved by vector representations in vector space

3.3. Summary

In this chapter the natural language processing tools that will be used to accomplish the goal of this work have been explained.

The preprocessing methods explained are used to remove unneeded information on the tweets to ease the processing of the information and lower the needed computing time. Also, to obtain the n-grams as they are represented in the word embeddings.

Different word embeddings models have been explained, as well as the different tasks that they can solve after learning the vector representations of the words.

In the next chapter, the process of building the multi-level ontologies needed as categories to classify tweets is explained.

4. Ontology building

The main aspects of smart cities that had to be analysed were about sustainability, economy and technology. There are other smart city aspects that could be analysed, such as governance, mobility, people or living. For this work, the focus was put on the three first mentioned aspects. The other mentioned smart cities aspects could be analysed in future work. Each of them needed an ontology, into which all the concepts related with those aspects had to be structured in a hierarchical manner.

In an ontology there are multiple levels of subclasses. The concepts that would be represented on the lower levels of the ontology had to be the more specific ones, and the concepts in the higher levels, the more general ones. This is why the root of each ontology was selected to be the aspect itself, because it is the most general concept of the ontology.

A thing that had to be considered when building the ontologies is that not all word embeddings are able to manage out-of-vocabulary words. Because it is not feasible to tune the concepts so that they are on the vocabularies of all the word embeddings, concepts were checked to be in the pretrained word2vec model GoogleNews-negative300. (Google Code, 2013)

To check the concepts appeared on the word embedding and that the meaning they had was the one it was intended to had, the 20 most similar words according to the embedding were looked at. For instance, in the following figure, the 20 most similar words to the word artificial intelligence in the word embedding vector space are shown.

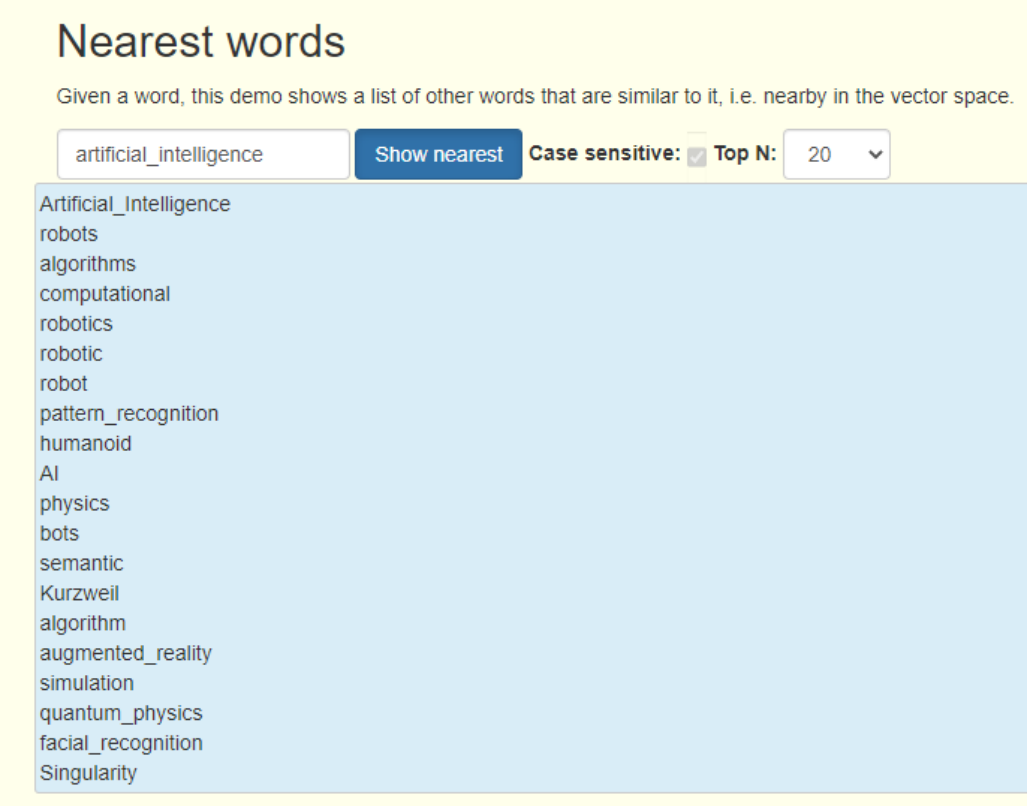

Figure 6: Top 20 nearest words to artificial intelligence

Some modifiers were also introduced in the concepts, allowing more flexibility for the classification of the tweets. They are the following ones.

- **Same concept modifier**: If multiple words are considered to refer to the same concept, they could be added in the ontology as comma separated values. The tweets would be compared to the different words, and if it is classified to any of those, it would be classified to that concept. There are no limits in the number of concepts that can refer to the same concept.
- **String-matching modifier**: If a word has to be compared by string matching instead of using the word embeddings comparison, the string-matching modifier can be used. To do so, a word has to start with the "*" symbol. This can be useful, for instance, in concepts that are not in the word embeddings vocabulary. An example can be in the concept "Smart_City", which is an important concept to have in the ontologies. As it is not contained in the word embeddings model used, in the ontology the word "*Smart_City" can be introduced to perform string-matching.
- **Disable-processing modifier**: If a concept has to be compared as it is on the ontology without any processing applied to it, the disable-processing modifier can be used. It is applied by adding the symbol "+" at the beginning of the concept. This can be useful for instance, when acronyms or abbreviations are introduced as concepts. This is important because some word embeddings take into account the capitalization of the words on the training data. The vector representations of "AI" (Artificial Intelligence) and "ai" are not similar when measuring the cosine similarity of the obtained vectors.

The three modifiers can be combined in order to obtain the desired comparison for the concepts on the ontologies.

Finally, another consideration taken into account when building the ontologies was that concepts present on the ontology had to be prioritized over string-matching concepts. Moreover, as much as it was possible, the sub concepts were tried to be as similar as possible to their parent concept in terms of cosine similarity.

Following, the three ontologies created for sustainability, economy and technology can be seen.

4.1. Economy ontology

In this ontology, economy concepts related with smart cities are represented. One of the concepts it includes is technological innovation. It has been included in this ontology instead of the technology one because it mainly contains products or services that are offered to the people or that they can benefit from them. The concepts IoT, iPhone_App, PPP, 3P and P3 have the disable-processing modifier because they lose its meaning if the casing is changed.

Finally, the concepts Collaboration, Partnership, Joint Venture and Alliance are considered the same as their vector representations are really close in the vector space. This would imply classifying the same tweets to all the four concepts. The same applies to Marketing and Branding.

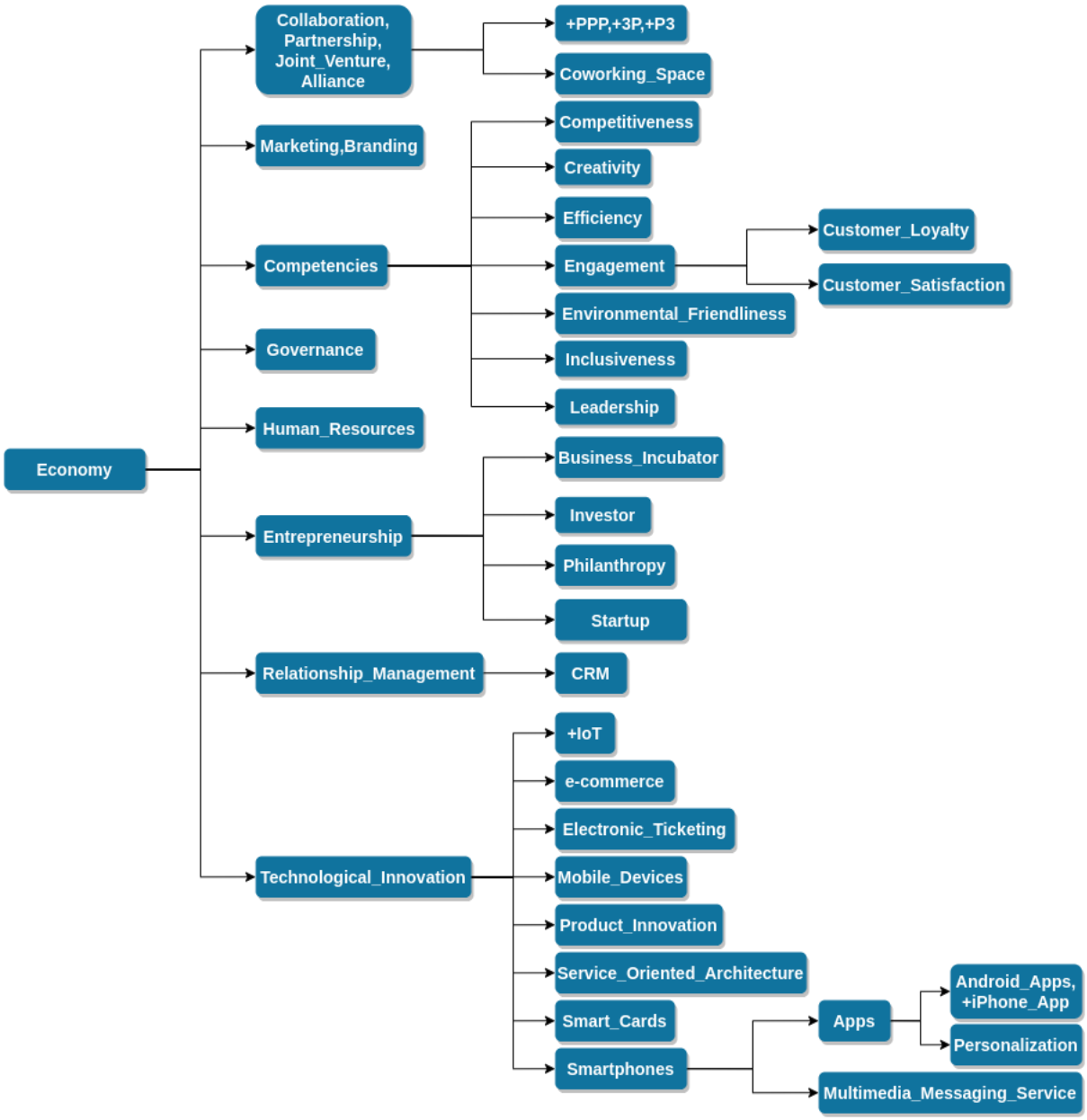

Figure 7: Economy concepts Ontology

4.2. Technology ontology

In the technology ontology, all the concepts are related mainly with innovative technology concepts. Some of them had to be added to be string-matched, as they were not present on the word embeddings, but they are very important concepts. They are Machine learning, Big data, Deep learning, and Wifi or Wi-fi. The Smart city concept also had to be added to be string-matched as it is also not in the word embedding.

For the Artificial Intelligence concept, it has also assigned the term AI with the disable-processing modifier, as it is its acronym and another casing on the acronym would not mean artificial intelligence.

Figure 8: Technology concepts Ontology

4.3. Sustainability ontology

For the sustainability concepts, they are all related with topics which regards the citizens of smart cities.

The concepts of climate change and global warming are grouped as the same concept because of their similarity on the word embedding. If they were considered as different concepts, most tweets classified on them would be equal.

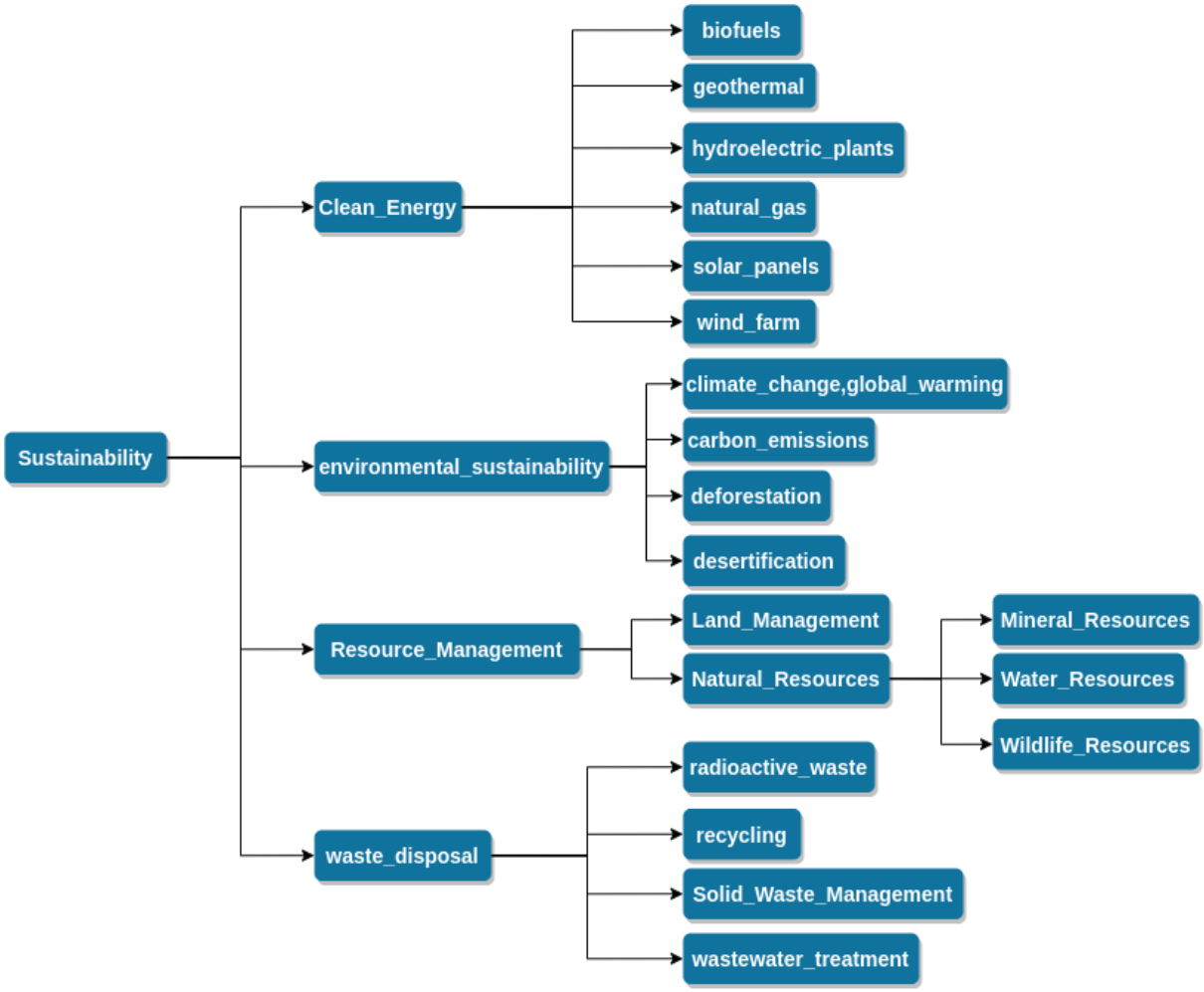

Figure 9: Sustainability concepts Ontology

4.4. Summary

The process followed to build the ontologies needed for the tweet's classification has been explained in this chapter. The different considerations that had to be taken into account when building the multi-level ontologies have also been explained.

Some special modifiers were also created in order to improve the classification of the tweets.

Finally, the three created ontologies for the smart city aspects economy, technology and sustainability are shown and some decisions taken in the process, commented.

In the following chapter, the tools explained in the previous chapter and the ontologies created in this one will be used for a proposal of a new system for tweet classification.

5. Methodology

In this section it is explained how the proposed system has been built and the tools and the resources that have been used in order to classify tweets in predetermined categories that are represented in multi-level ontologies.

5.1. Tweet classification in ontology concepts

The classification of a tweet into concepts from ontologies is divided into two main steps. First, the preprocessing of the tweets and the preprocessing of the concepts of the Ontology. Second, the classification of the processed tweets into the concepts of the processed Ontology. In the following figure, the main steps can be seen.

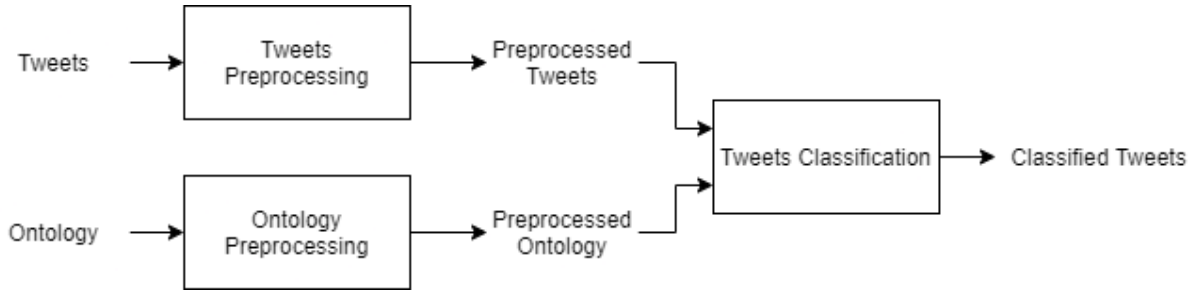

Figure 10: Tweets classification main steps

Some considerations had to be taken into account when designing these steps. For instance, some word embeddings have different vector representations depending on the capitalization of the words.

Moreover, word embeddings contain n-grams as if they were single words. To represent them as a single word, the blank spaces between the different words of an n-gram are represented as underscores. For instance, the n-gram “artificial intelligence” that is composed by the words “artificial” and “intelligence” would be represented in a word embedding as “artificial_intelligence”.

Finally, the approach that was wanted to be used to compare the tweets with the ontology concepts, was based on a word by word comparison, because the ontology concepts are words. As has been said, n-grams are considered as words, so they are treated as if they were single words.

In the following subsections, each of these steps is explained in detail.

5.1.1. Tweets preprocessing

The same preprocessing is applied to all the tweets that have to be analysed. The first step is removing the stop words and numbers of the tweet, because they do not provide any useful information regarding the tweet classification using word embeddings.

The second step is tokenizing the tweet in words. This step is divided into two sub steps. First, the tweet is tokenized by blank spaces. Second, the tokens that are hashtags, that is, start

with the '#' symbol, are segmented using a word segmenter because the multiple words they may contain are not separated.

Then, the n-grams have to be joined as single words. To find them, either n-gram chunking or an n-gram collocation model can be used to obtain the n-grams on the tweet. The n-grams that are found, are joined by using the underscore symbol as if they were single words.

Finally, the tokenized words resulting from the previous step are expanded to take into consideration that the word embeddings representations for a word can change depending on its capitalization because of the data with which has been trained.

The words expansion works the following way. If a word is all upper-case, it is not expanded, as it is highly probable that the word is an acronym or an abbreviation, and modifying it might result in a word that does not exist or has a different meaning. Otherwise, the word is transformed to lower-case. Then, another transformation is applied to the word. The first letter of the word is transformed to upper-case. If the word is an n-gram, the first letter of all the words that form the n-gram are transformed to upper-case. These two transformations are the ones that are used to perform the classification.

The first four steps are performed as they have been explained in the chapter 3, and the fifth as it has been explained in this chapter. Next, there is an example with all the tools applied to an example tweet.

Tweet: *On Tomorrow in #Aberdeen - What is Climate Change and why should I care?*

Step	**Preprocessing result**
Remove stop words	*On Tomorrow #Aberdeen - What Climate Change I care*
Word tokenization	*On, Tomorrow, #Aberdeen, -, What, Climate, Change, I, care*
Word segmenter	*On, Tomorrow, Aberdeen, -, What, Climate, Change, I, care*
n-gram chunking/collocation model	*On, Tomorrow, Aberdeen, -, What, Climate_Change, I, care*
Tweet expansion	*on, tomorrow, aberdeen, -, what, climate_change, I, care* *On, Tomorrow, Aberdeen, -, What, Climate_Change, I, Care*

Table 7: Tweet preprocessing example

Finally, the algorithm for the preprocessing of the tweets is the following one.

Algorithm 1: Tweet preprocessing

```
preprocessTweet(tweet):
    tweet = removeStopWordsAndNumbers(tweet)
    tokenizedTweet = tokenize(tweet)
    tokenizedTweet = hashtagSegmentation(tokenizedTweet)
    tokenizedTweet = ngramCollocation(tokenizedTweet)
    expandedTweetTokens = expandTweetTokens(tokenizedTweet)
    return expandedTweetTokens
```

5.1.2. Ontology concepts preprocessing

The preprocessing of the ontology concepts consists on managing the different modifiers they might contain and expand the concepts. The ontology concepts that are n-grams are already formatted using the underscore symbol as the separator between the words composing the n-gram. No preprocessing related to n-grams is needed.

If multiple words refer to the same concept, they will be separated using the comma symbol. All these words will be applied the same process.

First, if the word contains the string-matching modifier, it is transformed into lower-case and it is marked to be compared with the tweet by string-matching. Second, if the word contains the disable-processing modifier, no further processing is applied to the word and it is marked to be compared using word embeddings. In both of the previous cases, the modifiers are removed from the word.

If the word does not contain any of these modifiers and does not contain any letter in upper-case, it is marked to be compared using word embeddings. Otherwise, it is transformed to lower-case and both the original word and the lower-cased one are marked to be compared using word embeddings.

In the following table, there are some examples to the preprocessing applied to different ontology concepts.

Ontology concept	Preprocessing result
Entrepreneurship	Entrepreneurship entrepreneurship
*Machine_Learning	machine_learning
+IoT	IoT
Artificial_Intelligence,+AI	Artificial_Intelligence artificial_intelligence AI

Table 8: Ontology concepts preprocessing example

Finally, the algorithm for the preprocessing of the ontology concepts is the following.

Algorithm 2: Ontology concepts preprocessing

```
preprocessConcept(concept):
    conceptResult = {}
    for word in tokenize(concept, ‘,’):
        if containsStringMatching(word):
            concept[‘matching’].append(lower(removeModifier(word)))
        else if containsDisableProcessing(word):
            concept[‘embedding’].append(removeModifier(word))
        else:
            concept[‘embedding’].append(word)
            lowerWord = lower(word)
            if lowerWord != word:
                concept[‘embedding’].append(lowerWord)
    return conceptResult
```

5.1.3. Tweets and concepts comparison

The classification of the tweets is the result of a word by word comparison between the tweet's words and concepts obtained in the previous steps. This also works for the n-grams, because in the preprocessing of the tweets they have been found and joined using underscores as they are in the word embedding.

For the concepts that have to be string-matched with the tweet, two comparisons are made. One checks if the concept is in the tweet. The other, splits the concept and the tweet n-grams separating the multiple words containing underscores, and then performs the string-matching. This second comparison is only done if no matching is found in the first one, and allows to match the strings in case the n-gram finding method had not found all the n-grams properly. In both comparisons, all the words are transformed to lower-case, and if a concept is string-matched with a word of a tweet, a similarity of 1 is set to the comparison.

For concepts that have to be compared using the word embedding, first, the vector representations of the word and the concept in the word embedding are obtained. Then, their cosine similarity is obtained. The result of the cosine similarity ranges from -1 to 1, being -1 that the two vectors have no similarity, and 1 that they have maximum similarity.

If any of the similarities obtained in the comparison of a concept with all the words of a tweet is higher than a threshold that is given as a parameter, that tweet is marked as being similar to that concept. Setting a proper threshold value is important, because having a low threshold would result in too many wrongly classified tweets, and a high threshold in too many ontology related tweets not being classified.

This process is repeated for all the concepts, it does not stop when a concept with a cosine similarity higher than the threshold is found. After obtaining all the concepts that are related with a tweet, they are checked. Each tweet can be classified to multiple concepts, but they must not belong to the same branch of the ontology, and the check discards the more general concepts while preserving the more specific ones.

To exemplify the similarity values assigned to the words depending on the concepts, the table next shows some similarity values. The similarities are obtained from the GoogleNews-Negative300 word embedding.

Word	**Concept**	**Similarity**
recycling	recycle	0.812
robotics	Artificial_Intelligence,+AI	0.538
minerals	mineral_resources	0.633
environmentally_friendly	environmental_friendliness	0.611

Table 9: Similarity values examples

Moreover, in the table next, the checking of concepts given a list of all matches can be seen. If multiple concepts belong to the same branch of the ontology, just the more specific is kept.

Matching concepts	**Selected concepts**
Sensors, GPS_Tracking	GPS_Tracking
Drones, Artificial_Intelligence,+AI, *Machine_Learning	Drones, *Machine_Learning
wind_farm, environmental_sustainability, sustainability	wind_farm, environmental_sustainability

Table 10: Concept checking example

Finally, following there is the corresponding algorithm.

Algorithm 3: Tweets and concepts comparison

```
classifyTweets(concepts, tweets, wordEmbedding, threshold):
    results = {}
    for tweet in tweets:
        labels = []
        similarity = []
        for concept in concepts:
            for match in concept['matching']:
                if stringMatches(tweet, match):
                    similarity.append(1)
            for word in tweet:
                for embed in concept['embedding']:
                    similarity.append(embedding.cosineSimilarity(embed,
word))
            if any(threshold > sim for sim in similarity):
                labels.append(concept)
        results[tweet] = checkConcepts(labels)
    return results
```

5.2. Summary

In this chapter the proposed system to classify tweets from smart cities Twitter accounts to the concepts from the created ontologies is explained. First, the overview of the system, and afterwards each of the steps is detailed. The NLP tools explained in chapter 3 have been used to perform the steps, and examples of all the steps of the classification process are also provided.

In the next chapter, the proposed system will be used to obtain experimental results.

6. Experimental results

In this section experimental results are obtained analysing the tweets from different smart cities with the proposed system. First, the smart cities are selected and their tweets retrieved. Second, the n-gram finding method is selected. Third, the proposed system is evaluated using different word embeddings. Finally, the results obtained from the system for the selected smart cities Twitter accounts are analysed.

6.1. Selection of smart city Twitter accounts and data retrieval

To select the smart cities which had to be analysed, different rankings with top city destinations and top smart cities were looked for.

The ranking of smart cities selected (Giffinger, Fertner, Kramar, Pichler-Milanovic, & Meijers, 2007) ranks European smart cities. It includes 70 cities which are ranked in different categories, Smart Economy, Smart People, Smart Governance, Smart Mobility, Smart Environment, Smart Living and a global ranking.

Then, the Twitter accounts for those cities had to be searched. Some cities had multiple accounts related with the city. They were managed by different entities, such as the city council, the tourist office of the city, etc.

Once all the existing Twitter accounts for the cities, including multiple of them for a city if they existed, were found, a few of them had to be selected. In order to analyse the tweets from the accounts and know how many tweets each of them had, they were retrieved. To do so, by using libraries with access to the tweets, the date, tweet and tweet identifier were retrieved and stored in a different CSV file for each smart city Twitter account. The only processing done on the retrieved tweets was to remove the URLs.

The library used to retrieve the tweets is GetOldTweets3. It allows obtaining tweets according to the criteria set, including the username, date, top tweets, near a location, etc. It has been used instead of the official Twitter API because of multiple reasons. First, it does not have the limitations of the Twitter API. Their API does not allow for instance to retrieve tweets older than a week. Second, you can filter the tweets before retrieving them, so they do not have to be filtered afterwards. Finally, it can be used from the command line or as a Python library which adds more flexibility when retrieving the tweets. To remove the URLs from the tweets, a regular expression has been used, which is the following one *(?:(?:https|http)://)?\w[\w-]*(?:\.[\w-]+)+\S*'*.

The criteria to determine if a Twitter account was useful for the analysis was checking the number of tweets it had, the date of the last tweet and the language of the tweets. The Twitter accounts with less than a hundred tweets, without current tweets and using English in a few or no tweets, were discarded. In the following table, the 23 Twitter accounts from 21 different cities left after discarding the ones not fulfilling the beforementioned conditions are shown.

City	Twitter account	Number of tweets
Aarhus	@VisitAarhus	1497
Aberdeen	@AberdeenCC	25486
Bruges	@Visit_Bruges	2983
Cardiff	@VisitCardiff	15382
Cork	@corkcitycouncil	5175
Eindhoven	@EindhovenCity	917
Gent	@visitgent	6984
Graz	@VisitGraz	296
Innsbruck	@InnsbruckTVB	1294
Kaunas	@VisitKaunas	231
Kosice	@VisitKosice	141
Leicester	@Leicester_News	30347
Ljubljana	@visitljubljana	6617
Luxembourg	@luxembourginfo	3387
Miskolc	@VisitMiskolc	105
Portsmouth	@portsmouthtoday	7398
	@visitportsmouth	2421
Salzburg	@salzburg_info	3563
Tampere	@VisitTampere	1033
	@SmartTampere	474
Timisoara	@timisoara2021	124
Turku	@VisitTurku	4634
Zagreb	@zagreb_tourist	5249

Table 11: First selection of smart cities Twitter accounts

The final step consisted on manually checking the tweets of the remaining Twitter accounts to see if their tweets focus on topics related to smart cities or not. Moreover, the cities with two Twitter accounts, if both of them were smart cities related, just one of them could be kept. After this final selection, there were nine remaining smart cities Twitter accounts, which are from the cities Aberdeen, Cardiff, Cork, Eindhoven, Gent, Leicester, Ljubljana, Portsmouth and Tampere.

Those cities Twitter accounts are the ones that were selected to be analysed with the proposed approach. They can be seen in the table above with the coloured background.

6.2. Selection of n-gram finding method

Two methods to find n-grams have been explained on the chapter 3.1. They are n-gram chunking and n-gram collocation model. Either of them can be used on the proposed system.

The n-gram chunker was implemented using a grammar regular expression parser from the nltk library jointly with a chunker from the same library. The implementation of the n-gram collocation model was obtained from the genism library. It was trained using the text8 dataset (Mahoney, 2011), which contains the first 100000000 bytes of plain text from Wikipedia. To choose one of them to be used on the tests on following sections, a test was designed to check which of them is better on finding n-grams.

An n-gram test set had been build using the sample data from (N-grams data, n.d.). They provide a set of n-grams. From all of them, a 10000 n-grams were randomly selected. Moreover, the n-grams containing any stop word were filtered out, as stop words are not considered in the proposed system. After filtering them, 3361 n-grams were left for the test.

To test both n-gram finding techniques, they were provided with the words conforming n-grams. Then, it was checked if the methods were able to correctly detect the words provided as an n-gram. The result of the test is the following.

n-gram chunking	**n-gram collocation model**
597 / 3361 (17.76%)	638 / 3361 (18.98%)

Table 12: n-gram finding methods test results

From this test, it can be said that the n-gram collocation model is slightly better than n-gram chunking. To further check that its results will be good to find n-grams despite the low percentage of correctly detected n-grams, the ones that appear on the ontologies have also been manually tested. The results are the following ones.

N-gram	n-gram collocation model	Is correct?
solar panels	solar_panels	Yes
artificial intelligence	artificial_intelligence	Yes
natural resources	natural_resources	Yes
climate change	climate_change	Yes
global warming	global_warming	Yes

Table 13: n-gram collocation model with ontology n-grams

The model has been able to correctly detect all the n-grams that are present on the created ontologies. It should perform correctly with the tweets that have to be analysed.

6.3. Selection of word embeddings

The proposed system has to be tested to check if it is classifying properly or not the Smart City tweets in the concepts defined in the ontologies. In order to check it and evaluate its performance with different word embeddings, a test set was built in the following way.

First, the retrieved tweets of the 9 Smart City Twitter accounts were classified using the system. Then, for each city 125 tweets selected randomly were selected. From the 125 tweets, 50 of them were selected from all the available tweets, and 25 of the ones classified in each of the aspects, sustainability, economy and technology. It was allowed to choose repeated tweets, because those Twitter accounts have multiple tweets that are equal or so similar that after preprocessing them are equal. Also, if for a determined aspect the system has less than 25 tweets classified, all of them are selected. Finally, the 1109 randomly selected tweets were manually classified for each of the aspects, sustainability, economy and technology, obtaining 3 test sets with 1109 tweets each.

The tweets have been selected this way because a big amount of the tweets is not related to any of the evaluated aspects of the Smart Cities. Thus, randomly selecting the tweets from all the available ones would not select a representative number of tweets related with the Smart Cities aspects.

After building the test sets and obtaining the results given by the build system, those had to be compared to gain more insight about the results given by the system. The first metric computed was a class-wise confusion matrix, that is, obtaining a confusion matrix for each of the concepts. The confusion matrix has the count of true negatives at (0, 0), false negatives at (1, 0), true positives at (1, 1) and false positives at (0, 1).

The other metrics that were computed are the precision, recall, f1-score and support for each of the concepts, and the total averages. Being tp the true positives, fp the false positives and

fn the false negatives, the precision is computed as $\frac{tp}{tp+fp}$, the recall as $\frac{tp}{tp+fn}$ the f1-score the weighted harmonic mean of the precision and recall were both are equally important, and the support the number of occurrences of the class in the ground truth.

When $tp + fp == 0$ or $tp + fn == 0$ precision and recall are undefined respectively. In those cases, the value returned by default is 0.

For the total averages, different types of averaging used in multi-class classification are computed, which are the following ones:

- **Micro**: the metrics are computed globally by counting the total true positives, false negatives and false positives.
- **Macro**: the metrics are computed for each label and their unweighted mean is found. Label imbalance is not taken into account.
- **Weighted**: the metrics are computed for each label and their average weighted by support is found. It is an alteration to macro so that label imbalance is taken into account. A consequence of this is that the f1-score could not be between precision and recall.
- **Samples**: compute the metrics for each instance and find the average.

Then, some pre-trained word embeddings created using different approaches were selected to test the system. They were the following ones:

- **Google News N300**: word2vec model with negative sampling trained with part of the Google News dataset (about 100 billion words). It contains 300-dimensional vectors for 3 million cased words and phrases. (Google Code, 2013)
- **Fasttext Common Crawl**: fasttext model trained with 600 billion tokens (subword information) obtained from Common Crawl. It contains 300-dimensional vectors for 2 million words. (Mikolov, Grave, Bojanowski, Puhrsch, & Joulin, 2017)
- **GloVe Common Crawl**: GloVe model trained with 840 billion tokens obtained from Common Crawl. It contains 300-dimensional vectors from 2.2 million cased words. (Pennington, Socher, & Manning, 2014)

To load and obtain the vector representations of this word embeddings, the genism library has been used. It allows to load all kinds of embeddings and to obtain the cosine distance between the vector representations of two words independently of the word embedding chosen. Moreover, they can be loaded using mmap, so they can be mapped into memory instead of having to be completely loaded into it. Apart from reducing the word embedding load time, it allows using them on computers with less memory. To load the fasttext models, the fasttext library can also be used. The only difference with the genism one is that it needs less memory when loading the word embeddings model.

The threshold that is needed by the program had to be set for each of the word embeddings, as its value depends on them. To set them, the values in the range [0.4, 0.9] were tested in

increments of 0.1 and the threshold with better accuracy when compared with the manually classified ground truth was selected. The resulting thresholds are the following ones.

Word embedding model	Google News N300	Fasttext Common Crawl	GloVe Common Crawl
Threshold	0.6	0.9	0.6

Table 14: Word embeddings thresholds

6.4. Evaluation of the system

The proposed system has been tested using the different word embedding models to see which one performs better in the tweet classification task with smart city related concepts. To do so, each of the results obtained with the proper thresholds for each word embedding model, has been compared with the ground truth obtaining the metrics that have already been explained.

There are two ways to obtain the results from the proposed system. It can be executed as an interactive program which asks for the needed options and parameters, and it can also be executed as a command line program in which the needed parameters are given in the same command. Both methods produce as a result an Excel file containing the tweet classification.

To obtain the needed metrics, the ground truth has been formatted as if it was a result file obtained from the program. Then, both the ground truth and the result of the proposed system are loaded, and the metrics are computed using the sklearn.metrics functions multilabel_confusion_matrix and classification_report.

Following, the results obtained for each of the ontologies can be seen in a table, and in a visualization for easier analysis of the results.

6.4.1. Sustainability results

Model	Metric	Micro Average	Macro Average	Weighted Average	Samples Average
Google News N300	*Precision*	0.96	0.68	0.98	0.97
	Recall	0.98	0.74	0.98	0.98
	F1-score	0.97	0.7	0.98	0.97
Fasttext Common Crawl	*Precision*	0.82	0.23	0.79	0.83
	Recall	0.82	0.17	0.82	0.83
	F1-score	0.82	0.17	0.79	0.83
GloVe Common Crawl	*Precision*	0.82	0.15	0.79	0.82
	Recall	0.82	0.15	0.82	0.82
	F1-score	0.82	0.14	0.8	0.82

Table 15: Metrics of the Sustainability results

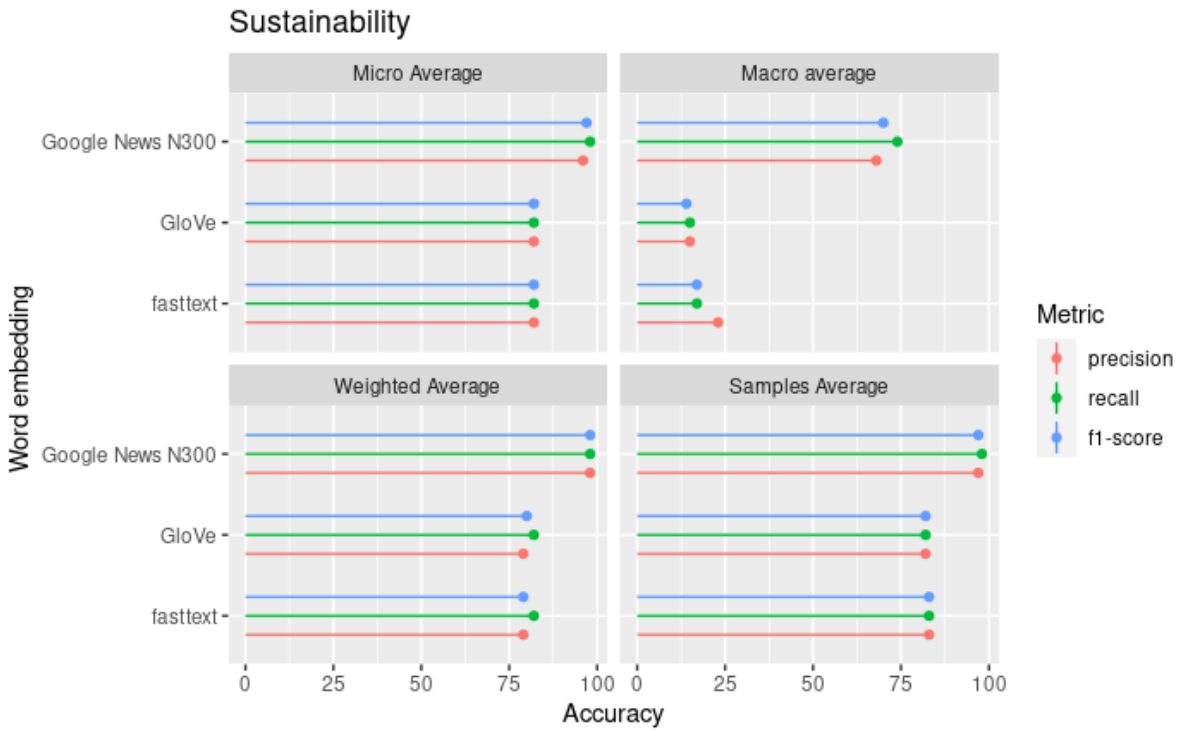

Figure 11: Visualization of the metrics of the Sustainability results

6.4.2. Economy results

Model	Metric	Micro Average	Macro Average	Weighted Average	Samples Average
Google News N300	*Precision*	0.9	0.8	0.93	0.93
	Recall	0.94	0.92	0.94	0.95
	F1-score	0.92	0.83	0.93	0.93
Fasttext Common Crawl	*Precision*	0.86	0.47	0.79	0.86
	Recall	0.84	0.38	0.84	0.86
	F1-score	0.85	0.41	0.8	0.86
GloVe Common Crawl	*Precision*	0.69	0.24	0.82	0.8
	Recall	0.81	0.51	0.81	0.82
	F1-score	0.75	0.28	0.8	0.81

Table 16: Metrics of the Economy results

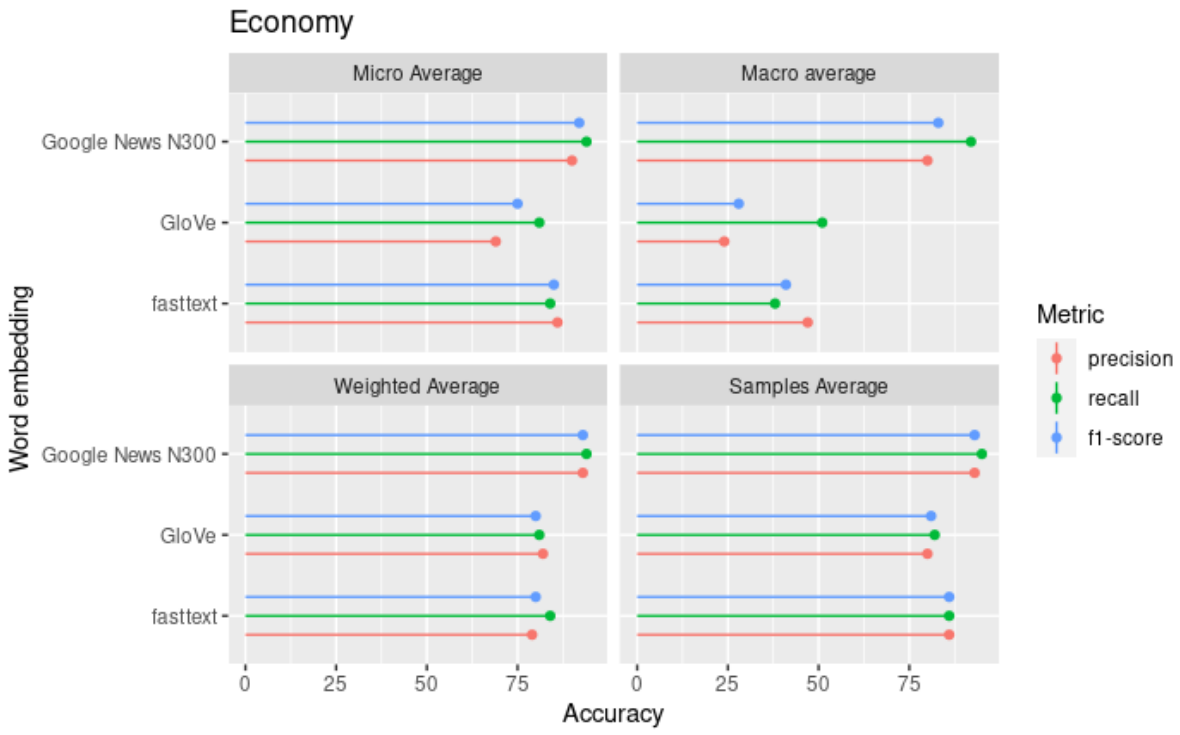

Figure 12: Visualization of the metrics of the Economy results

6.4.3. Technology results

Model	Metric	Micro Average	Macro Average	Weighted Average	Samples Average
Google News N300	*Precision*	0.96	0.67	0.96	0.96
	Recall	0.95	0.66	0.95	0.95
	F1-score	0.95	0.65	0.95	0.95
Fasttext Common Crawl	*Precision*	0.84	0.49	0.76	0.84
	Recall	0.83	0.39	0.83	0.83
	F1-score	0.83	0.42	0.78	0.83
GloVe Common Crawl	*Precision*	0.8	0.33	0.72	0.81
	Recall	0.8	0.36	0.8	0.8
	F1-score	0.8	0.33	0.75	0.8

Table 17: Metrics of the Technology results

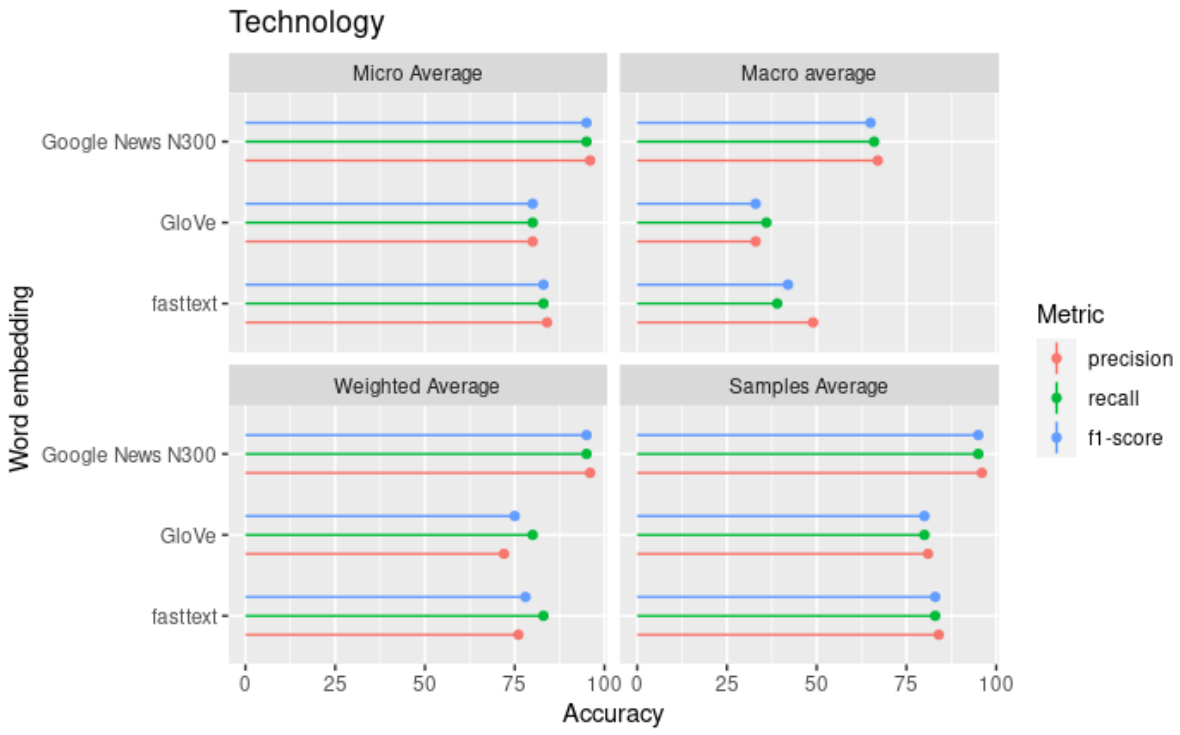

Figure 13: Visualization of the metrics of the Technology results

6.5. Analysis of the results

Some conclusions can be extracted from the obtained results. First, that overall, the word embedding model that achieves the best results for this tweet classification task is the Google News N300. The other two word embedding models have similar results, but the ones obtained by Fasttext Common Crawl are a bit better than the ones obtained by GloVe Common Crawl.

Second, that overall, the classification made by the system is quite accurate, as it can be seen on the micro, weighted and samples averages. This is because those averages take into account that the test set is unbalanced, as some concepts have numerous tweets classified, whereas some others have very few tweets.

The lower values of the macro averages, which do not take into account the imbalance on the test set, indicates that the system is not so accurate with the concepts which have few tweets assigned, which tend to be more specific concepts. This could be expected, because the concepts with fewer tweets are more specific, and the word embeddings tested are not able to correctly classify them because of their lack of representation of the polysemy of words. Moreover, the word embedding models are pretrained for general use, fine-tuning them for this specific task would probably improve its results.

To perform a further analysis of the obtained results, the ones obtained using the GoogleNews-negative300 have been used, because they are the best ones from the tests performed. In the following table the number of tweets classified for each of the accounts can be seen.

City	Retrieved tweets	Labels assigned to Sustainability	Labels assigned to Technology	Labels assigned to Economy
Aberdeen	25486	1192 (4.66%)	764 (3%)	761 (2.96%)
Cardiff	15382	49 (0.32%)	274 (1.78%)	179 (1.16%)
Cork	5175	172 (3.3%)	167 (3.23%)	345 (6.53%)
Eindhoven	917	19 (2.07%)	60 (6.53%)	287 (26.82%)
Gent	6984	17 (0.24%)	31 (0.44%)	74 (1.06%)
Leicester	30347	809 (2.66%)	1389 (4.57%)	821 (2.69%)
Ljubljana	6617	83 (1.25%)	66 (1%)	100 (1.51%)
Portsmouth	7398	376 (5.06%)	240 (3.24%)	223 (3%)
Tampere	474	33 (6.89%)	276 (53.8%)	148 (28.14%)

Table 18: GoogleNews-N300 results summary

To know if the smart cities which make a higher number of tweets about the three analysed aspects correspond to their achievement of that aspects as a smart city, they can be compared with their position in a ranking of smart cities.

The ranking made by (Giffinger, Fertner, Kramar, Pichler-Milanovic, & Meijers, 2007) evaluates smart cities from different aspects, including sustainability and economy. Following, the positions obtained out of a total of 70 smart cities by the analysed cities are shown.

City	Position in Sustainability	Position in Economy
Aberdeen	67	10
Cardiff	60	13
Cork	66	2
Eindhoven	39	6
Gent	48	19
Leicester	64	3
Ljubljana	3	8
Portsmouth	63	7
Tampere	12	29

Table 19: Ranking of smart cities

From both previous tables, it can be seen that the best cities on the ranking are not the ones which make a bigger number of tweets in respect to the total tweets they have. For instance, in sustainability, Ljubljana is the 3rd smart city in the ranking, but just a 1.25% of the tweets they have made are about this aspect. By contrast, the worst city on sustainability on the ranking, Aberdeen, which is the 67th, has 4.66% of tweets related to sustainability, almost four times the ones from Ljubljana.

In the economy aspect, the best results in the ranking were obtained by Cork, which even though has a big number of tweets related, 6.53%, is the 3rd smart city in number of tweets about economy. For the city of Tampere, even though has the worst results on the ranking in the economy aspect, has the greater number of tweets about it 28.14%. It is also the smart city with more tweets related with sustainability and technology. Its total number of tweets is the lower one, and most of them are related to the analysed smart city aspects.

In general, it can be said that there is no correlation between the position of a smart city in the smart city rankings and the number of tweets they make about the analysed smart city aspects.

Finally, the labels assigned for each of the aspects and smart cities have been summarized in tables. They are analysed following. The green, orange and yellow cells represent the higher, second and third higher number of labels assigned for each smart city respectively.

Concept/City	Aberdeen	Cardiff	Cork	Eindhoven	Gent	Leicester	Ljubljana	Portsmouth	Tampere		Total
Sustainability							1		1		2
Clean_Energy	6		2			12		9			29
Hydroelectric_plants	6										6
natural_gas	104	11	3		1	16		30	1		166
biofuels	7	1				1		3	1		13
geothermal	1							1	1		3
solar_panels	5		3	9		4		4			25
wind_farm	3			1							4
Resource_Management											0
Land_Management			1								1
Natural_Resources	8	1	7			10					26
Wildlife_Resources	23	9				43		2			77
Water_Resources	3		12			1		1			17
Mineral_Resources	1	2									3
environmental_sustainability	42	15	25	8	6	8	74	7	10		195
climate_change	41		20		1	51	1	14	2		130
global_warming	41		20		1	51	1	14	2		130
carbon_emissions	39	2	17		6	73	2	17	14		170
deforestation	9	2	4			3					18
desertification											0
waste_disposal	181	3	26	1	1	166	2	56			436
wastewater_treatment	37		1						1		39
recycling	623	3	28		1	357	1	218			1231
radioactive_waste			3								3
Solid_Waste_Management	12					13	1				26
Total	1192	49	172	19	17	809	83	376	33		

Table 20: Sustainability labels assignment

For sustainability, the most recurring concept is recycling, followed by waste disposal and environmental sustainability. Even though these concepts are the ones most tweeted about, most of the other ones are also quite tweeted, such as natural gas, carbon emissions, or climate change and global warming. Just a few of them are barely tweeted or not tweeted at all. For instance, desertification or wind farm. In the case of sustainability, it is normal that has just 2 tweets assigned, because it is the concept at the root of the ontology and most likely, concepts are assigned to more specific concepts instead.

The cities which make more tweets about sustainability are Aberdeen, Leicester and Portsmouth. Even though these cities are the ones with a higher number of tweets, the one which has a higher percentage of tweets related with sustainability is Tampere, with a 6.89% of tweets about this aspect from the total 474 tweets analysed. In most of them, they mention most of the concepts, excepting Eindhoven, Gent, Ljubljana and Tampere, which are the ones with fewer tweets about sustainability, and they are focused in certain specific concepts. The concepts which are tweeted by most of the cities are environmental sustainability, carbon emissions, waste disposal and recycling.

Concept/City	Aberdeen	Cardiff	Cork	Eindhoven	Gent	Leicester	Ljubljana	Portsmouth	Tampere		Total
Technology	33	5	4	29	5	10		3	12		101
Smart_City			7	4					176		187
Artificial_Intelligence,AI			1	1		1			15		18
Drones	4	1	1	3		3			2		14
Predictive_Analysis	1					1			5		7
Wireless_Connectivity	4	2	1			1			2		10
Broadband_Connectivity	12	4	4			54		43	2		119
Infrastructure	64	1	6			11	1	3	2		88
Robotics	9	5	2	11	7				20		54
Machine_Learning									6		6
Monitoring	72	2	10			35		14	3		136
Sensors	2			1					2		5
Deep_Learning									1		1
Data_Science									1		1
Traffic_Congestion	352	11	53			432	16	80	4		948
Privacy	1								2		3
Lidar									5		5
Biometrics									1		1
Cybersecurity	3	1	3			1		1	1		10
User_Experience									1		1
Chatbot									3		3
Smart_Grid	1							1	1		3
Social_Networking	161	210	52	6	11	662	18	75	9		1204
Big_Data	3			1							4
Virtual_Reality	2	2		4				1			9
Cloud_Computing	17	6	11		5	25	2	9			75
Modeling	2	1	3								6
ICT			2								2
Accessibility	1	2				3	5				11
Wi-fi,wifi	15	21			3	121	24	10			194
Intelligence_BI	1										1
Information_Technology	4		7			29					40
Total	764	274	167	60	31	1389	66	240	276		

Table 21: Technology labels assignment

In the case of technology, something similar to sustainability happens. The top mentioned concepts are social networking, traffic congestion and Wi-fi. Most other topics are also tweeted, but much less, such as monitoring, smart city, or broadband monitoring. Just a few of them are barely tweeted or not tweeted at all, like intelligence BI, biometrics or data science.

The cities which have more tweets about technology are Leicester, Aberdeen and Tampere. The last one is the third one with respect to the tweets on technology, but it is the first one with respect to the percentage of tweets assigned to technology. A total of 53.8% of its tweets are about technology, which is much higher than Eindhoven, which is the second one in percentage of tweets on technology with a 6.53%. The majority of tweets from all the cities are focused on the top mentioned concepts listed before. Despite this fact, most cities mention a big variety of concepts, but with much fewer tweets than the top mentioned ones. The cities Eindhoven, Gent, Ljubljana and Portsmouth are an exception, having less variety of concepts on their tweets.

Concept/City	Aberdeen	Cardiff	Cork	Eindhoven	Gent	Leicester	Ljubljana	Portsmouth	Tampere		Total
partnership	161	20	96	2	17	64	10	24	11		405
alliance	62	8	21			46	1	10			148
joint_venture	56	8	18			43	1	7			133
customer_satisfaction	66	6	10			114	1	11			208
smartphones	28	12	3	1	5	91	5	32	2		179
economy	36	5	11		1	54	1	2			110
technological_innovation	54	5	19	64	11	16	1	2	37		209
creativity	105	25	45	69	10	40	34	17	13		358
innovation	31	12	28	63	7	9	14	13	20		197
entrepreneurship	26	1	30	36		14	1	3	14		125
android_apps,iphone_app	33	56	5	3	15	280	20	52	15		479
social_entrepreneurship	2		5	2	1			5			15
engagement	36		17			4			2		59
efficiency	8		3	1		17		14	2		45
operational_efficiencies	3		1				1	8			13
marketing	4										4
leadership	10					2	2	1			15
governance	3		2			2		10	1		18
collaboration	6	1	1	10		1	1	3	9		32
branding	24	11	18	14	6	7	4	4	2		90
environmental_friendliness	1		2			2	1	2			8
crm	1										1
smart_cards	1					8		1			10
ppp,3p,p3	1										1
apps	2	1			1	1	1		1		7
philanthropy	1	1	1			2					5
inclusiveness		1	4					1			6
dynamism		6	2						1		9
coworking_space			1								1
startup			2	22		2		1	17		44
human_resources						2					2
personalization							1		1		2
Total	761	179	345	287	74	821	100	223	148		

Table 22: Economy labels assignment

Finally, in the case of economy, the results are a bit different to both previous aspects. The most tweeted concepts, android apps, iPhone app, partnership and creativity, are not so tweeted as in the previous aspects. Moreover, other concepts have a very similar number of tweets, such as customer satisfaction or innovation. There are also a few concepts with very few tweets, such as coworking space or human resources.

In this case, the cities with more tweets about economy are Leicester, Aberdeen and Cork. As in the previous aspects, despite not being the city with more tweets about it, Tampere is the one with higher percentage of them. It is the city with fewer tweets in total, but the one which has a higher percentage of tweets about the analysed aspects. A 28.14% of the tweets of Tampere are about economy, which is very similar to the 26.82% of Eindhoven, which is the second city in percentage of tweets about economy. The main focus for all the cities is very similar. The concepts which have more importance for each city are very similar. In general, the less important concepts are also mentioned by all the cities, but with much smaller number of tweets on them.

6.6. Summary

This chapter shows all the experimental results obtained from the proposed system. First, the smart cities to have their Twitter accounts analysed have been selected, and their tweets retrieved.

The method to be used to find the n-grams in the tweets has also been selected from the two possible methods, n-gram chunker and n-gram collocation model. The second one has been selected for its better results. Also, the word embeddings model to obtain the vector representations of the words to use in the proposed system has been selected. Multiple tests have been performed in order to find which pretrained model has better results to solve the intended task. After analysing the results, the word embedding based on word2vec GoogleNews-Negative300 has been selected.

After selecting the n-gram finding method and the word embedding to be used, the results obtained from the classification of the retrieved tweets with the proposed system have been shown. These results have also been analysed at the smart city aspect level, as well as at the ontology concepts level.

In the following chapter, the conclusions of this word and future word that could be done related to it are described.

7. Conclusions and future work

The objective of this work was to develop a methodology that can automatically analyse a set of tweets retrieved from some selected smart cities Twitter accounts. The methodology had to be able to detect if the retrieved tweets were related with smart city concepts and classify them accordingly.

To do so, three different multi-level ontologies have been created. They cover the sustainability, technology and economy aspects of smart cities. Moreover, they are prepared to be used with word embeddings, as they include n-grams formatted the same way as they can be found in the word embeddings.

Different word embedding models have also been analysed. They have been tested for the purpose to be accomplished in this work, and the one which best suited it among all of them is the word2vec model GoogleNews-negative300. The other word embeddings tested, GloVe and fasttext also have good results, but not as good as the ones of GoogleNews-negative300.

Finally, the created ontologies with the selected word embedding have been used in the proposed system, achieving successfully the main goal of the work and all the sub-objectives.

Nine different smart cities have been selected to be analysed, and their tweets have been retrieved. They all appear in a smart city ranking (Giffinger, Fertner, Kramar, Pichler-Milanovic, & Meijers, 2007), which has allowed to analyse the relation between their positions in the different ranked aspects and the number of tweets related to those aspects.

The results have shown that there is no correlation between having a big number of tweets in an aspect and having a good position in that ranking. Some smart cities such as Tampere have a low number of tweets, but they all are about smart cities aspects, whereas smart cities with a much bigger number of tweets such as Cardiff, barely speak about those aspects. Each of the cities should be analysed individually in order to know how they communicate about the three analysed aspects.

For the results on the specific concepts which are talked about, it can be concluded that most cities focus on communicating about the same concepts. Despite this, in general all the cities communicate about all the concepts found on the ontologies.

As future work, the following objectives could be considered.

- Extend the existing ontologies with more concepts.
- Create new ontologies about other smart cities aspects such as governance or transport.
- Test other word embedding models, including other pretrained models, or train and fine tune a new model.
- Overcome the lack of polysemy the used word embeddings have.
- Find a way to use the whole tweet for the classification, instead of analysing it word by word.

Bibliography

Alkhammash, E. H., Jussila, J., Lytras, M. D., & Visvizi, A. (2019). Annotation of smart cities Twitter micro-contents for enhanced citizen's engagement. *IEEE Access, 7, 116267-116276.*

BIrd, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. . *O'Reilly Media, Inc.*

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proceedings of German Society for Computational Linguistics*, pp. 31-40.

Dabiri, S., & Heaslip, K. (2018). Dabiri, S., & Heaslip, K. (2018). Twitter-based traffic information system based on vector representations for words. *arXiv preprint arXiv:1812.01199.*

Dai, X., Bikdash, M., & Meyer, B. (2017). Dai, X., Bikdash, M., & Meyer, B. (2017, March). From social media to public health surveillance: Word embedding based clustering method for twitter classification. *In SoutheastCon*, pp. 1-7.

Encalada, L., Boavida-Portugal, I., Ferreira, C. C., & Rocha, J. (2017). Identifying tourist places of interest based on digital imprints: Towards a sustainable smart City. *Sustainability (Switzerland), 9 (12)*.

Fand, A., Macdonald, C., Ounis, I., & Habel, P. (2016). Using word embedding to evaluate the coherence of topics from twitter data. *In Proceedings of the 39th International Association for Computing Machinery's Special Interest Group on Information Retrieval conference on Research and Development in Information Retrieval*, pp. 1057-1060.

Farajidavar, N., Kolozali, S., & Barnaghi, P. (2017). A deep multi-view learning framework for city event extraction from twitter data streams. *arXiv preprint arXiv:1705.09975.*

Giffinger, R., Fertner, C., Kramar, H., Pichler-Milanovic, N., & Meijers, E. (2007). *europeansmartcities.* Retrieved from http://www.smart-cities.eu/download/smart_cities_final_report.pdf

Globerson, A., Chechik, G., Pereira, F., & Tishby, N. (2007). Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research, Vol. 8*, pp. 2265-2295.

Google Code. (2013). Retrieved from https://code.google.com/archive/p/word2vec/

Huertas, A., Moreno, A., & Tran, H. M. (2019). Which destination is smarter? Application of the (SA) 6 framework to establish a ranking of smart tourist destinations. *International Journal of Information Systems and Tourism (IJIST)*, 4(1), 19-28.

Jabreel, M., Huertas, A., & Moreno, A. (2018). Semantic analysis and the evolution towards participative branding: Do locals communicate the same destination brand values as DMOs? *PloS one*, 13(11), e0206572.

Jabreel, M., Moreno, A., & Huertas, A. (2017). Semantic comparison of the emotional values communicated by destinations and tourists on social media. *Journal of destination marketing & management*, 6(3), 170-183.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kummitha, R., & Crutzen, N. (2017). How do we understand smart cities? An evolutionary perspective. *Cities (67)*, pp. 43-52.

Lalicic, L., Huertas, A., Moreno, A., & Jabreel, M. (2019). Which emotional brand values do my followers want to hear about? An investigation of popular European tourist destinations. *Information Technology & Tourism*, 21(1), 63-81.

Lalicic, L., Huertas, A., Moreno, A., & Jabreel, M. (2020). Emotional brand communication on Facebook and Twitter: Are DMOs successful? *Journal of Destination Marketing & Management, 16, 100350*.

Lebret, R., & Collobert, R. (2013). Word emdeddings through hellinger PCA. *arXiv preprint arXiv:1312.5542*.

Levy, O., & Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. *Proceedings of the eighteenth conference on computational natural language learning.*, pp. 171-180.

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, pp. 2177-2185.

Li, Q., Shah, S., Liu, X., Nourbakhsh, A., & Fang, R. (2016). Tweetsift: Tweet topic classification based on entity knowledge base and topic enhanced word embedding. *In Proceedings of the 25th Association for Computing Machinery International on Conference on Information and Knowledge Management*, pp. 2429-2432.

Mahoney, M. (2011, 9 1). Retrieved from http://mattmahoney.net/dc/textdata.html

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*, pp. 3111-3119.

N-grams data. (n.d.). Retrieved from https://www.ngrams.info/iweb.asp

Norvig, P. (2009). Natural language corpus data. *Beautiful data*, pp. 219-242.

Ontotext. (2016). Retrieved from https://tag.ontotext.com/about/

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532-1543.

Pereira, J., Pasquali, A., Saleiro, P., & Rossetti, R. (2017). Transportation in Social Media: an automatic classifier for travel-related tweets. *In EPIA Conference on Artificial Intelligence*, pp. 355-366.

Roig, A. H., Moreno, A., & Tran, H. M. (2018). Evaluation and ranking of smart tourist destinations. *In XII Congreso Internacional de Turismo y Tecnologías de la información y las comunicaciones*, (pp. pp. 47-63). Universidad de Málaga (UMA).

Tran, H. M., Huertas, A., & Moreno, A. (2017). 6: A new framework for the analysis of smart tourism destinations. A comparative case study of two Spanish destinations. *Actas del Seminario Internacional Destinos Turísticos Inteligentes: nuevos horizontes en la investigación y gestión del turismo*, (pp. pp. 190-214). Universidad de Alicante.

Villena-Román, J., Cobos, A. L., & Cristóbal, J. (2014). TweetAlert: Semantic Analytics in Social Networks for Citizen Opinion Mining in the City of the Future. *In UMAP Workshops.*