NETHIC: A System for Automatic Text Classification using Neural Networks and Hierarchical Taxonomies

Andrea Ciapetti[1], Rosario Di Florio[1], Luigi Lomasto[1], Giuseppe Miscione[1], Giulia Ruggiero[1] and Daniele Toti[1,2][a]

[1]*Innovation Engineering S.r.l., Rome, Italy*

[2]*Department of Sciences, Roma Tre University, Rome, Italy*

Keywords: Machine Learning, Neural Networks, Taxonomies, Text Classification.

Abstract: This paper presents NETHIC, a software system for the automatic classification of textual documents based on hierarchical taxonomies and artificial neural networks. This approach combines the advantages of highly-structured hierarchies of textual labels with the versatility and scalability of neural networks, thus bringing about a textual classifier that displays high levels of performance in terms of both effectiveness and efficiency. The system has first been tested as a general-purpose classifier on a generic document corpus, and then applied to the specific domain tackled by DANTE, a European project that is meant to address criminal and terrorist-related online contents, showing consistent results across both application domains.

1 INTRODUCTION

With the increasing use of social networks and with the digitalization of governmental structures, computer scientists are facing new challenges and needs. Usually, official communications and documentations are stored in the form of electronic textual documents. The rest of the personal communications exchanged by single individuals is represented by chat messages, tweets, e-mails, blog entries, etc. This leads to an increased volume of textual information that may consequently bring about increasing confusion and hinder the effectiveness of the communication itself. Understanding the subject category each data item falls into and the topics discussed has become paramount for an effective management and analysis of this deluge of information. Indeed, during the latest years, significant effort and considerable resources have been spent to satisfy this need within the context of governmental and commercial projects (Dalal and Zaveri, 2011). One of the potential techniques to be used in this regard is the automatic text classification, which falls into the category of supervised machine learning tasks. Such a process is meant to automatically assign a set of pre-defined classes by using a machine learning technique (Sebastiani, 2002). This paper describes NETHIC, an automatic text classification system based on a hierarchical taxonomy and artificial neural networks (ANNs). Taxonomies represent knowledge in a structured and human-readable manner. Their hierarchical structure enables an efficient and automated content classification. (Wetzker and et al., 2008). Artificial neural networks, on the other hand, have some interesting properties that make this family of machine learning algorithms very appealing when facing difficult pattern-discovery tasks. This combined approach is especially useful when a large amount of data is used during the training phase, and can be easily implemented in parallel architectures (*i.e.*, with multi-core processors or systems with dedicated GPUs). This may drastically reduce the processing time compared to other kinds of algorithms, while achieving similar results in terms of effectiveness (Hermundstad et al., 2011). In this work, the NETHIC system is detailed, showing how it displays a significant level of performance by using different taxonomies. First, a generic taxonomy is used in order to obtain a general-purpose text classifier. Then, a specific taxonomy to tackle domain-specific texts and concepts from the DANTE Horizon 2020 project is introduced, so that it could be used to classify documents dealing with terrorist and criminal activities as well, the latter being the very objects of the DANTE project itself.

This paper is structured as follows. In Section 2,

[a] https://orcid.org/0000-0002-9668-6961

Ciapetti, A., Di Florio, R., Lomasto, L., Miscione, G., Ruggiero, G. and Toti, D.
NETHIC: A System for Automatic Text Classification using Neural Networks and Hierarchical Taxonomies.
DOI: 10.5220/0007709702960306
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 296-306
ISBN: 978-989-758-372-8

the DANTE project is introduced and related work is discussed. Section 3 introduces the building blocks of the system in terms of the taxonomies and datasets used. Section 4 describes the actual solution created for text classification, in terms of its system architecture and classification process. Section 5 shows the experimentation carried out for the system and its performance in terms of accuracy and efficiency. Finally, Section 6 concludes the paper and provides pointers to future work.

2 CONTEXT AND RELATED WORK

The problem of the automatic classification of textual documents is one of the important tasks solved by text mining methods. A number of diverse applications of text classification were reported in literature, ranging from subject categorization (Sebastiani, 2002), analysis of sentiment of reviews or opinions, to authorship recognition of documents (Wang and Manning, 2012; Koppel and Winter, 2014), etc. Standard methods of text classification represent documents with usually high-dimensional feature vectors, and then train classifiers such as SVM, Naive Bayes, k-NN (Vidhya and Aghila, 2010; Wang and Zhao, 2012), etc. Although several ways of representing documents with feature vectors were proposed (*e.g.* (Forman, 2003)), a commonly used approach consists of building feature vectors which represent (potentially weighted) frequencies of selected words or collections of words (bigrams, n-grams, phrases) that appear in subsequent documents. These approaches can be broadly named as bag-of-words methods. In the traditional bag-of-words approach the keywords are filtered from training data. Usually, some Natural Language Processing methods can be involved such as: Segmentation, Tokenization, POS Tagging, Entity Detection, Relation Detection (Bird et al., 2009); these methods are broadly used in several general-purpose and/or domain-specific applications and solutions (Toti et al., 2012; Atzeni et al., 2011c; Atzeni et al., 2011a; Atzeni et al., 2011b; Toti and Rinelli, 2016). Creating such objects from text can give a lot of information about its content. The appearance and frequencies of specific tokens or entities are used as a basis for bag-of-words model. However, the number of this kind of objects can be very large. Therefore, methods to reduce dimensionality of data are needed, for instance TF-IDF, PCA, LDA, SVD, t-SNE (Li et al., 2015; Lamar et al., 2010; Kowsari et al., 2017), etc. to keep only the most important words for classification. In this work, the purpose is to investigate the feasibility of a conceptually different approach, by representing documents with feature vectors, and training classifiers. Two different ways of representing sequences of words for training are used: with a simple encoding of words, and with the Word2Vec method which represents words in vector space (Kumar et al., 2015). An initial verification of this approach is provided based on a collection of Wikipedia articles representing subject categories, with 500 articles per category (for a general-purpose taxonomy). The idea behind this paper is to show how to build an effective text classifier by using state-of-the-art machine learning methods and tools and what issues can arise during this procedure. NLTK (Natural Language Processing Toolkit) algorithms have been used here for tokenization. For feature extraction, instead, a different approach has been adopted: rather than encoding the frequencies of keywords, the words from sentences have been directly transformed into sequences of encoding vectors and have been used for training deep learning methods. This approach shares a common factor with probabilistic models such as n-grams, conditional random fields and other Markov-models, which also use sequences and are based on the probability of appearance of specific words. This approach has been then verticalized onto a specific domain given by the DANTE project. DANTE (Detecting and ANalysing TErrorist-related online contents and financing activities - project id: 700367) is an ongoing innovation project funded by the European Community under the Horizon 2020 grant framework, whose purpose is to deliver efficient and automatic solutions for data mining, analytics, as well as an integrated platform to detect, retrieve, and analyze huge amounts of heterogeneous and complex multimedia and multi-language terrorist-related contents from both the Surface and the Deep/Dark Web. Its ultimate goal is to discover, analyze and monitor potential terrorist-related activities and people, with a focus on online fund raising, propaganda, training and communication activities.

3 NETHIC'S BUILDING BLOCKS: TAXONOMIES AND DATASETS

This section describes the building blocks of the system, by detailing the structure of the hierarchical taxonomies needed for the system to work and the datasets used.

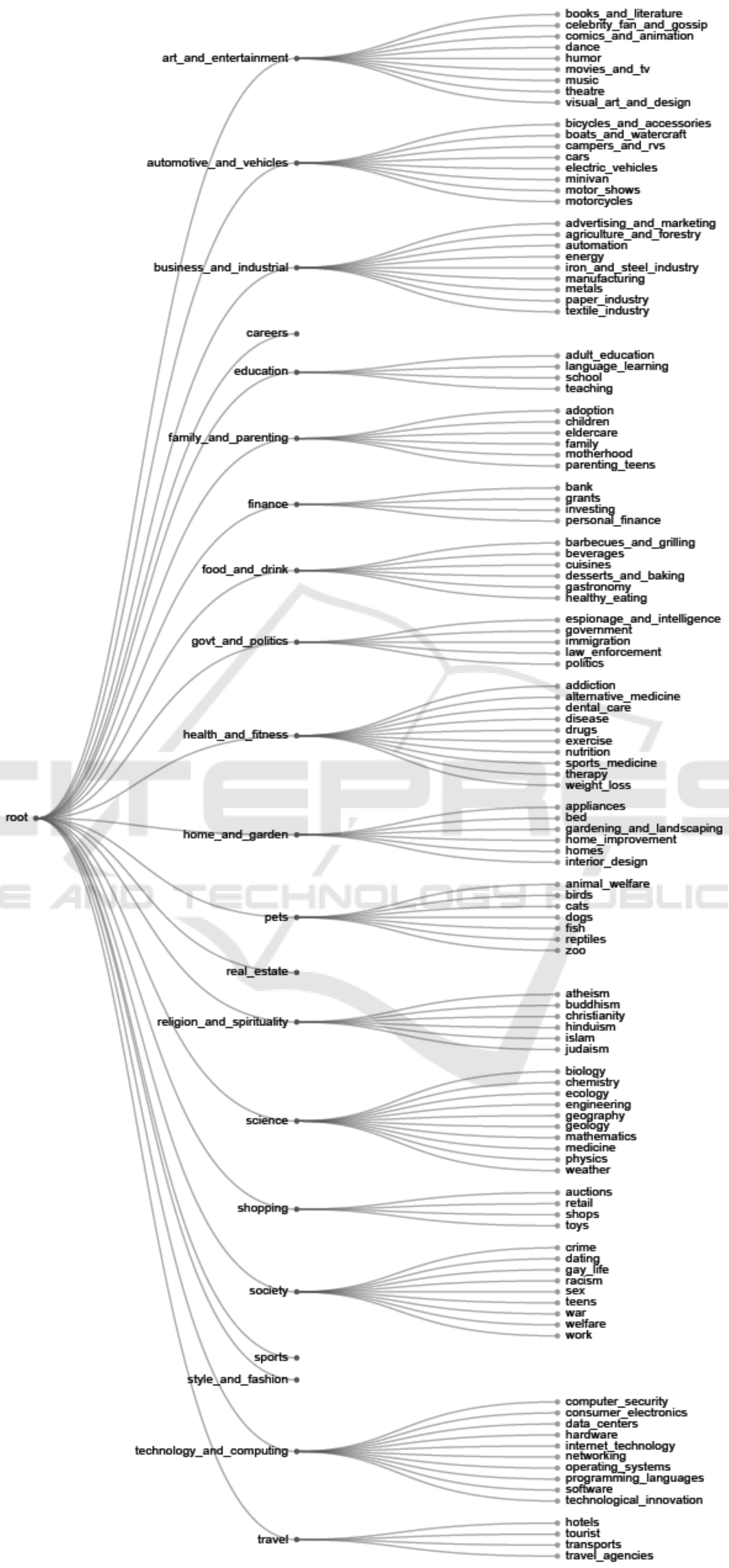

Figure 1: General-purpose taxonomy.

3.1 Hierarchical Taxonomies

Taxonomies play a core role within NETHIC, because a taxonomy is needed to assign the pre-defined classes for the system's neural networks and to build and organize the required dataset (Dalal and Zaveri, 2011). Taxonomies, as they stand, are tree-like structures meant as effective means to put a certain degree of order upon unstructured information and thus to more easily access it. Logical categories are used to classify such information in order for users to conveniently browse this information. They are also often employed in conjunction with search and retrieval tools, with the purpose of restricting the scope of the search (Lamont, 2003); as such, they enable users/systems to focus on a specific area of interest (Tang et al., 2006).

In NETHIC two taxonomies have been defined. The first is a general-purpose taxonomy, and as such is able to cover a wide range of general topics; it is made up of a root node and 21 top-level child categories, 17 of which with a number of subcategories/leaves Figure 1. The second taxonomy is instead domain-specific and has been put in place to tackle the terrorist/criminal domain of the DANTE project; it includes a root node, 4 top-level child categories, and several sub-categories and leaves (Figure 2). By using the structure given by each taxonomy, two datasets have been built, with 57,304 and 12,512 documents, respectively. In Section 4.3, how these datasets have been used to build the training and test sets is described.

3.2 Dataset Construction

The main issue to be faced, when creating a modular, general and hierarchical classifier, is the availability of the datasets with which to train the underlying neural networks. In order to make up for this, a number of documents have been collected from Wikipedia (Alarcón et al., 2009), due to the latter being the largest data knowledge archive openly available on the Internet nowadays. Wikipedia's freely accessible APIs have been used to query it in order to gather the documents needed. This process is as follows. Firstly, Wikipedia's whole category graph, *i.e.*, the knowledge base's internal structure used to organize its massive amount of data, is queried. Then, from each of its categories, the list of subcategories and the documents belonging to it are retrieved. This results in a graph containing about 1.5 million categories. The subsequent step is to extensively use a word embedding model (Mikolov et al., 2013) to calculate the vector for each category, saving this information in a database. Using these vectors it is possible to weight the links between a parent category and their children using the inverse cosine similarity formula (Lahitani et al., 2016). By having a graph weighted according to the semantic value of its categories, it is then possible to take advantage of the properties of Dijkstra's algorithm (Chen, 2003). Therefore, starting from a category and using Dijkstra, documents can be collected from Wikipedia by following a semantic path. This process ends either when the desired number of documents are collected, or when a given category does not have any other subcategories. Finally, the hierarchical taxonomy mentioned in Section 3.1 is merged with the category graph, by mapping each leaf of the taxonomy to a Wikipedia category, creating a structure of folder and sub-folder that possesses the same structure of the taxonomy and where documents are contained only in the leaf folders. In 4.3 further details will be provided about how many documents are collected and how they have been divided into training and test datasets.

4 NETHIC'S ARCHITECTURE AND CLASSIFICATION MECHANISM

In this section NETHIC's architectural model is described, along with the motivation and the choices made to obtain a neural network hierarchy. The data pre-processing and training algorithms implemented to bring about the neural networks will be described as well.

4.1 Architecture

The general structure of NETHIC's architectural model is based on a neural network classifier, whose categories are the concepts contained in the hierarchical taxonomy shown in Section 3.1. A neural network hierarchy has been used to increase the system's accuracy for a single application domain. For example, in order to classify a paper that talks about *kitchens*, it may make sense to use a neural network trained only on documents focused on interior decoration and house supplies, instead of a more heterogeneous or too general neural network. In this way, a lot of unnecessary words will not be considered and the noise on the classification function will be reduced. In the proposed implementation, for each taxonomy concept, except for the leaves, a neural network is trained and a dictionary of words is built. In the first levels, the trained neural networks are characterized

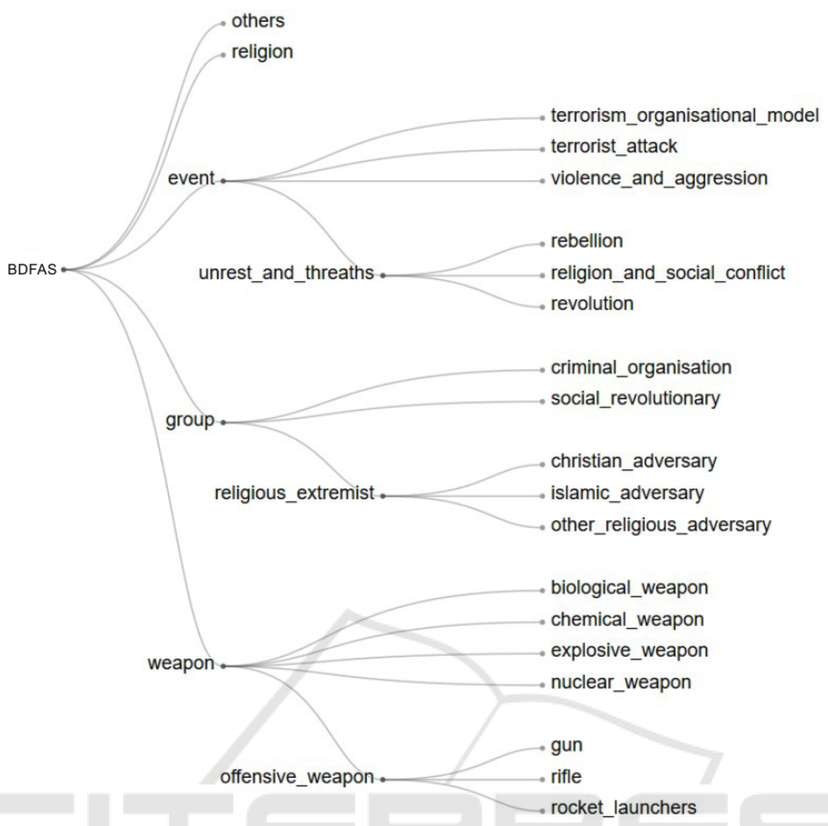

Figure 2: DANTE-related taxonomy.

by a horizontal view, which divides documents into global concepts like Sports, Science, Economy, Religion, whereas in the nested levels the neural networks tend to assume a more vertical division and classify the documents inside a specific category of the general concept (here: Sports), for example Basketball, Combat Sports, Football, Swimming, Tennis, Volleyball, etc. Thus, the classification function used in the neural networks, located at different levels in the hierarchy, is trained with a vocabulary and a set of words, which have a different logical structure and granularity. The upper levels are trained with generic words, similar to general "concepts", and the vocabulary used does not contain the complete glossary of words, associated with the context. Descending towards the deeper levels, instead, the need to differentiate between semantically similar concepts increases. Therefore, the glossary used in the training process is structured with more specific words, because the classification function now needs to acquire further knowledge of the specific area of interest, in order to effectively classify the textual contents. As such, the classification process follows an iterative approach — very suitable for systems based on ANNs — descending to the deeper and more specific levels of classification. This allows the system to avoid semantic errors with words that belong to several conceptual areas like, for example, the word *tail* (used for describing both a body part of an animal and a particular data structure in computer science). This also means that in some cases the iteration should be stopped earlier, before reaching the deepest levels, when dealing with generic texts, as it is better explained in the following paragraphs. These observations are very important, since they are the basis of the following optimizations: (1) Noise reduction by using appropriate glossaries of words for training; (2) Smaller dimensional space to build vectors from documents (described in the next section) that increases computational performance. As shown in Figure 3, the main roles in this architectural structure are played by the Pre-Processing and Categorizer modules. Starting from a textual content to be classified, each neural network uses the associated dictionary to transform the text into a numeric vector, evaluates the most appropriate category for the classification (at the current stage of iteration at this level of the hierarchy) and calls the next neural network associated with it, which iteratively performs the same operation. To improve the performances, neural networks and dictionaries are loaded at the beginning and kept in memory until the end of the process. This choice may involve the usage of conspicuous amounts of memory.

4.2 Data Pre-processing

The first step of the classification mechanism is represented by data pre-processing. Several studies show different approaches (Sebastiani, 2002) for the text mining process. In NETHIC, standard steps have been chosen to transform non-structured text into structured data. First of all, text is cleaned from stop-words and numbers; afterwards, stemming and lemmatization algorithms are applied on the remaining words. The outcome of this process is a number of cleaned sentences. After this cleaning step, the next one is to transform unstructured data in structured form. To do that, the *CountVectorizer* function of the *scikit-learn* Python library is used to generate a vector of word counts for each document. Dictionaries are created in the training step, with a set of words collected from the corpus used to train the model, so that it is possible to ensure that a particular neural network is linked with a correct dictionary that contains the main words for the specific context.

4.3 Training

As explained in the previous subsection, in order to obtain a high accuracy level, a crucial part of the classification process is represented by the training process and by the construction of appropriate glossaries at the different levels of the hierarchy. For the general-purpose taxonomy, a vectorized dataset is obtained that contains 57,304 documents, about 500 documents for each leaf category of the taxonomy. The dataset has been split into training and test sets with a ration of 95% and 5%, respectively, thus a *training-dataset* with 54,439 documents and a *test-dataset* with 2,843 documents have been produced. The subdivision uses the same proportion for each category. This is a good way to verify the model's accuracy. Since they are driven by the structure of the hierarchical tree of the taxonomy and by the number of its nodes, exactly 18 neural networks (17 for the categories that have subcategories, and 1 for the root node) have been trained. Since NETHIC's model has a hierarchical structure, while the dataset contains documents for leaf concepts only, a bottom-up approach to build intermediate datasets for training the neural networks becomes necessary. A document that belongs to a specific concept (*e.g.*, chemistry), intuitively belongs at the linked generic concept (*e.g.*, science): therefore, initially, at the *n-1* level, documents are used for the child concepts. However, in order to successfully train the neural networks for the highest levels, it is necessary to build a dataset with at least 500 documents for each concept. To do this, documents from sub-categories in the dataset are used. For example, to build the *science* dataset to train the *root* neural network, 50 documents for each sub-category are taken (*biology, chemistry, ecology, engineering, geography, geology, mathematics, medicine, physics, weather*).

Table 1: Configuration details.

Function	**Solver**	**Accuracy(%)**		
		Training	**Test**	**Model**
tanh	sgd	22.83	25.33	▨
identity	adam	97.31	83.68	76.74
identity	lbfgs	96.74	82.15	67.56
relu	lbfgs	97.02	80.52	68.65
logistic	sgd	18.13	18.48	▨
logistic	adam	96.66	83.58	77.87
tanh	lbfgs	96.87	80.76	66.16
relu	sgd	20.63	19.99	▨
relu	adam	97.31	83.32	77.06
tanh	adam	97.32	83.91	76.46
logistic	lbfgs	25.06	27.31	▨
identity	sgd	23.32	29.07	▨

Each neural network has only one hidden layer with 60 neurons since it has been observed by manual experimentation that this choice is the best in terms of accuracy and complexity. In order to choose the best settings, a combination of *function-solver* (featured in the *sklearn.neural_network.MLPClassifier* library) with highest accuracy has been selected. Activation function is used by the neurons to calculate the value to be sent to the next level. Solver is used for weights optimization. In Table 1, results with all combinations are shown, where each accuracy value is the average value of all neural networks' accuracy. *logistic-adam* pair was chosen because it returns the best accuracy for the *root* NN level, which in NETHIC's approach is the the most influential level. This configuration has resulted in good levels of performance with the data used for training. More than half of the distribution is over 96%, the lowest value is near 95% and the low variance as well indicates how the model is a good fit for the problem tackled. A similar process has been used for training the system with DANTE's domain-specific taxonomy (omitted here for brevity).

4.4 Algorithm to Build Paths

NETHIC's model is based on two main parameters:

- **cutoff** — This parameter is used by any single neural network to decide how many categories to consider. Its initial value is 0.8, and in each iteration it is decremented by 0.1. For a single classification process, categories are kept until the cutoff is reached;

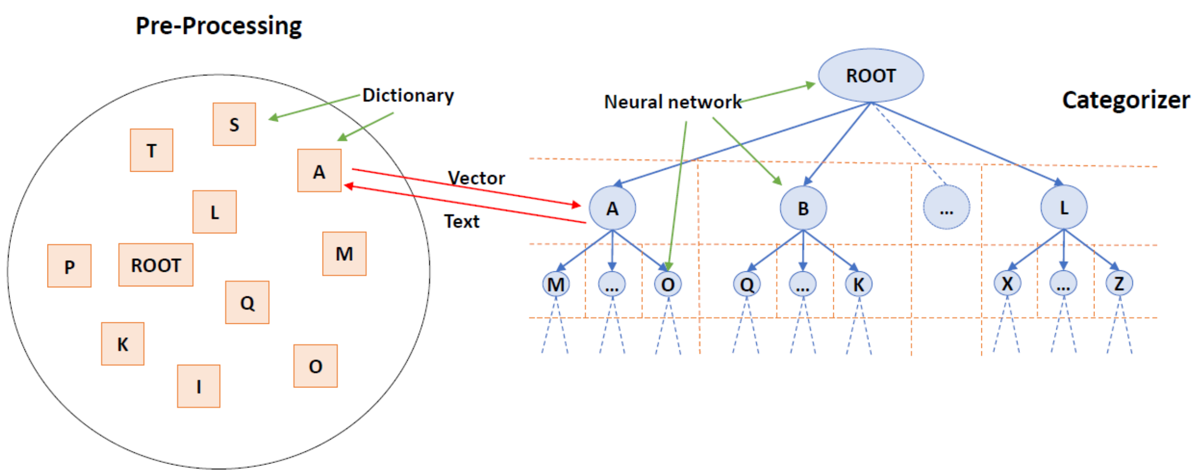

Figure 3: Architecture.

- **tolerance** — Represents the threshold to be reached to deem the set of the results produced as valid. This value has been set to 0.7, and for each occurrence the *currente_tolerance* is calculated as the average of the path scores and compared with *tolerance*.

For each returned path, the average between all the single scores for any corresponding category is computed. For example, given the following path: *P = C1/C2/C3* with its respective scores *SC1*, *SC2* and *SC3*, its corresponding path score is: *SP = (SC1+SC2+SC3)/3*. In Algorithm 1 the pseudo-code that chooses the categories to return for the next level of the hierarchy is shown. Basically, the system keeps considering categories until the probabilities returned by the current neural networks (a value between 0 and 1) reach a threshold. If after the first classification a good *current_tolerance* is obtained, it will consequently lead to a good classification; otherwise, if such a value is low, it means that there are paths with a low score. In this case, the cutoff is decremented to consider less categories in every neural network and a second classification iteration is run. In general, this algorithm allows the system to select the highest-level categories and concepts when the textual content examined contains only generic terms, whereas it is possible to select more detailed and low-level categories and concepts by examining texts that are very specific, technical or focused on a specific topic.

5 EXPERIMENTAL RESULTS

This section describes NETHIC's experimental results.

Algorithm 1: Classification.

```
1:  procedure GETRESULTS(cutoff, tolerance, textToClassify)
2:      cutoff ← 0.8
3:      tolerance ← 0.7
4:      results ← {∅}.
5:      while cutoff <= 0.4 AND current_tolerance < tolerance do
6:          current_tolerance ← 0
7:          raw_results ← {∅}.
8:          raw_results ← classification(cutoff, textToClassify)
9:          cutoff ← cutoff − 0.1
10:         results ← computeResults(raw_results)
11:         for i in results do
12:             current_tolerance ← current_tolerance + results[i]/max(results)
```

5.1 Accuracy and Performance with a General-purpose Taxonomy

To evaluate NETHIC's classification accuracy, tests have been performed that consider both the single neural networks and the general model that takes into account the whole general-purpose taxonomy (heretofore referred to as "global" model). In order to do this, a dataset with 2,843 documents has been used. Firstly, for each neural network trained, as previously stated, a small dataset has been used to test the 18 neural networks. To build datasets for categories deeper down into the hierarchy, the same training procedure has been used. For evaluating the accuracy, only the first category returned has been considered, given the fact that each neural network could return a variable number of category labels. The first thing to be noticed is that the model does not suffer from overfitting, *i.e.* training and test results show consistent levels of accuracy, since each neural network is apparently able to generalize well in terms of predictions against new data. The lowest results are returned by the *root* neural network, which represents an outlier in this regard. Given the high number of labels in this classification, an accuracy of 72.31%, considering only the first label returned, can be considered a satisfactory result. In order to better clarify this concept, the corresponding confusion matrix has been provided to highlight the causes of the errors. In the left side of Figure 4, the diagonal matrix is immediately apparent. By observing the high values out of the diagonal, the confusion is caused by semantically-close classes such as: *arts and entertainment* and *society*, or *technology and computing* and *business and industry*, or *health and fitness* and *food and drink*. In fact, classes semantically distant such as *sports* and *shopping*, or *style and fashion* and *automotive and vehicles* do not display confusion. Afterwards, all the neural networks have been merged together to classify textual contents by using the whole hierarchical structure. In this case it was chosen to return the first three paths/labels given by the classifier. The reason for this choice is that a textual content can belong to more than one concept or topic. In a hierarchy where the leaf concepts are 119, it is in principle correct to assign multiple labels. By using this assumption, the same 2,843 documents, earlier classified with a single real label, have been classified again, and an accuracy equal to 77.87% has been obtained. By looking at the global confusion matrix for these results (right side of Figure 4), it is possible to notice how the error is lower then in the previous results. In this case, in fact, there are less errors due to the semantic proximity. As proof of this, a general example is provided below.

To evaluate the performance based on the length of the text, the dataset has been split into two subsets: (without considering repetitions): (1) 1 - 100 words (1553 documents); (2) 101 - 200 words (1254 documents).

The pie charts in Figure 6 show a performance improvement when more words are used. Obviously if more words are used to build a vector, more words are found in the dictionaries, thus there will be more items in the vectors greater than 0.

General Example: *Racism in sports has been a prevalent issue throughout the world, and in particular racism towards African-Americans has been especially bad over the course of the history of sports in the United States and around the world. [...]*

Results: society/racism/: 0.5580656885761913; sports: 0.4927055063449122

In the dataset this document is marked as sports, whereas the first label returned by the classifier is *racism*, thus it can be deemed an error. If the correct label is to be found within the first three labels returned instead (in this specific case only two are returned), then the classification process can be deemed successful.

5.2 Results with a DANTE-specific Taxonomy

To perform the classification process on DANTE-related contents a different taxonomy, shown in Figure 2, has been used. Before analyzing the domain-specific contents, the system has been tested against Wikipedia documents. Results, shown in the box-plot of Figure 5, display a good overall performance. The greatest confusion is found at the top of the matrix between the similar concepts **group** and **event**. The global accuracy obtained for this model is 87% on 642 test documents. The single neural networks produced show good accuracy with values between 77% and 94%. All observations made for the global model are valid also in this case. Afterwards, to assess the effectiveness of the system on real, DANTE-related use cases, a specific dataset has been used for the test. Using DANTE-related contents (a collection of terrorist-related documents), a small document set made up of 20 documents has been built, 5 for each of the following categories: **Weapon, Other, Religion, Religious Extremist**. The main goal here is to detect criminal and terrorist contents (weapons, religious extremism) and distinguish them from harmless elements (other, religion). On this data the system has shown an accuracy equal to 95%. A couple of excerpts are presented below.

Religious Extremist Example: *The death of a single Muslim, no matter his role in society, is more grave to the believer than the massacre of every kāfr on earth. And while the Sharī'ah calls for the invasion of all kāfr lands, certainly the aggressors are dealt with before those nations not actively waging war against the Khilāfah. This is an obvious reality. Any disbeliever standing in the way of the Islamic State will be killed, without pity or remorse, until Muslims sufer no harm and governance is entirely for Allah. Brussels, the heart of Europe, has been struck. The blood of its vitality spilled on the ground, trampled under the feet of the mujāhidīn. Flames ignited years ago in Iraq have now scorched the battleground of Belgium, soon to spread to the rest of crusader Europe and the West. Paris was a warning. Brussels was a reminder. What is yet to come will be more devastating and more bitter by the permission of Allah, and Allah prevails.*

Results: event/violence_and_aggression/: 0.53;
group/religious_extremist/islamic_adversary/: 0.52;
event/unrest_and_threaths/religion_and_social_conflict/: 0.47

Weapon Example: *5.4 Fusogen (Nerve Gas) It can*

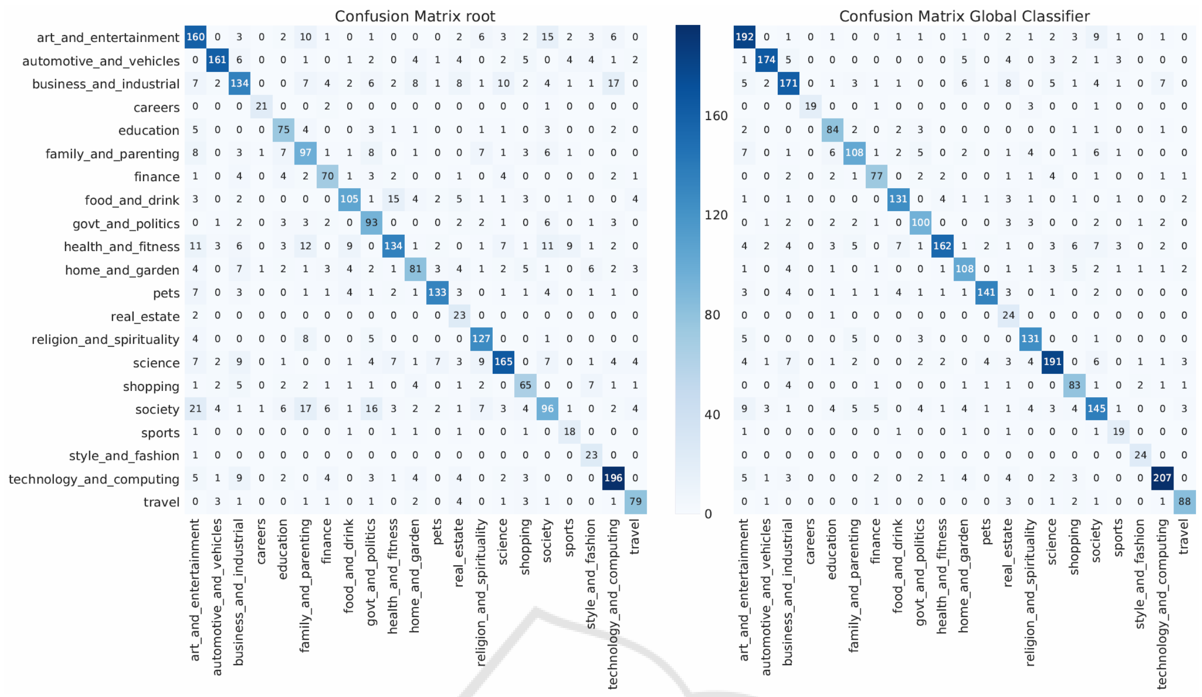

Figure 4: Confusion matrix for the root and the global neural network.

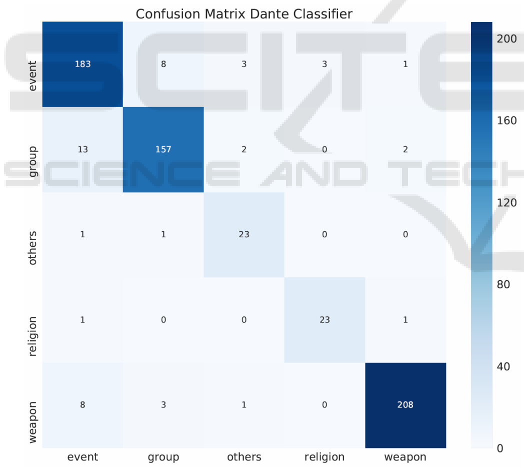

Figure 5: Confusion matrix for the DANTE model.

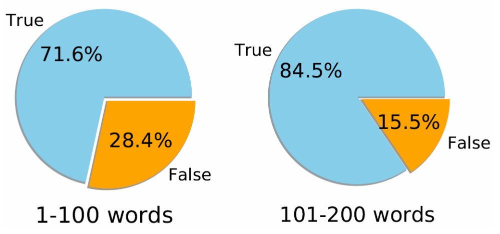

Figure 6: Test accuracy with different sizes.

be obtained by heating CCl4 with any metal. 5.5 Nitrous Oxide: N2O (Laughing gas) Can be obtained by heating Ammonium Nitrate (a well known fertiliser and explosive compound) between 250^o-260^o. In a closed room, your victims will laugh to death. [. . .]

Results: weapon/chemical_weapon/: 0.83

Religion Example: *"Who is a Jew?" is a basic question about Jewish identity and considerations of Jewish self-identification. The question is based on ideas about Jewish personhood, which have cultural, ethnic, religious, political, genealogical, and personal dimensions. Orthodox Judaism and Conservative Judaism follow the Halakha, deeming a person to be Jewish if their mother is Jewish or they underwent a proper conversion. Reform Judaism and Reconstructionist Judaism accept both matrilineal and patrilineal descent. Karaite Judaism predominantly follows patrilineal descent. Jewish identity is also commonly defined through ethnicity. Opinion polls have suggested that the majority of Jews see being Jewish as predominantly a matter of ancestry and culture, rather than religion. Ashkenazi Jews, being the most numerous Jewish ethnic division, have been the subject of numerous genealogical studies and have been found to be a distinct, homogeneous ethnic group.*

Results: religion/: 0.91

Other Example: *The Pet Travel Scheme ("PETS") is a system which allows animals to travel easily between member countries without undergoing quarantine. A Pet Passport is a document that officially records information related to a specific animal, as part of that procedure. The effect is to drastically speed up and simplify travel with and transport of an-*

imals between member countries, compared to previous procedures, if the regulations are followed.
Results: other/: 0.96

5.3 Hardware and Implementation

The hardware used to run the tests includes a Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz with 16 GB RAM. The actual classifier has been developed in Python with the *scikit-learn*, *CountVectorizer* and *Multi-layer Perceptron (MLP)* libraries used to build vectors and for creating the neural networks. The *pickle* library has been used to persist the neural networks and to load them in memory. In general, with this hardware the time taken to classify a document is less than 25 milliseconds.

6 CONCLUSION AND FUTURE WORKS

This paper presented NETHIC, a classifier for textual contents based on hierarchical taxonomies and neural networks. The results reported showed that this combined approach has several advantages for tackling the given task, including modularity, scalability and overall performance. The system has proved itself to be successful both on a general and a specific, terrorist and crime-related domain. As far as the latter is concerned, future developments may include the possibility of applying the classification process for enhancing other components of the DANTE project itself, especially related to the discovery of criminal and terrorist networks from social media via the analysis of the textual contents shared and discussed online. Additional improvements may also involve the introduction of deep learning algorithms in order to increase NETHIC's global accuracy, for example by introducing more hidden layers for each neural network and different activation functions (*e.g.* via frameworks like TensorFlow (Abadi et al., 2015)); this may lead to further refinements of NETHIC's classification mechanism.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alarcón, R., Sánchez, O., and Mijangos, V. (2009). Exploiting Wikipedia as a knowledge base: Towards an ontology of movies. 534:8–16.

Atzeni, P., Polticelli, F., and Toti, D. (2011a). An automatic identification and resolution system for protein-related abbreviations in scientific papers. In *Lecture Notes in Computer Science*, volume 6623 LNCS, pages 171–176.

Atzeni, P., Polticelli, F., and Toti, D. (2011b). Experimentation of an automatic resolution method for protein abbreviations in full-text papers. In *2011 ACM Conference on Bioinformatics, Computational Biology and Biomedicine, BCB 2011*, pages 465–467.

Atzeni, P., Polticelli, F., and Toti, D. (2011c). A framework for semi-automatic identification, disambiguation and storage of protein-related abbreviations in scientific literature. In *Proceedings - International Conference on Data Engineering*, pages 59–61.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit*. O'Reilly.

Chen, J.-C. (2003). Dijkstra's shortest path algorithm. *J. of Form. Math.*, 15.

Dalal, M. K. and Zaveri, M. (2011). Automatic text classification: A technical review. 28.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305.

Hermundstad, A., Brown, K., Bassett, D., and Carlson, J. (2011). Learning, memory, and the role of neural network architecture. 7.

Koppel, M. and Winter, Y. (2014). Determining if two documents are written by the same author. *J. of the Assoc. for Information Science and Technology*, 65:178–187.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., and Barnes, L. E. (2017). Hdltex: Hierarchical deep learning for text classification. 2.

Kumar, A., IRsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. In *CoRR*.

Lahitani, A. R., Permanasari, A. E., and Setiawan, N. A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International Conference on Cyber and IT Service Management*, pages 1–6.

Lamar, M., Maron, Y., Johnson, M., and Bienenstock, E. (2010). SVD and clustering for unsupervised POS tagging. In *ACL*.

Lamont, J. (2003). Dynamic taxonomies: keeping up with changing content. *KMWorld*, 12.

Li, J., Chen, X., Hovy, E., and Jurasky, D. (2015). Visualizing and understanding neural models in NLP. In *CoRR*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47.

Tang, L., Zhang, J., and Liu, H. (2006). Acclimatizing taxonomic semantics for hierarchical content classification. In *Proc. of the 12th ACM SIGKDD*, pages 384–393.

Toti, D., Atzeni, P., and Polticelli, F. (2012). Automatic Protein Abbreviations Discovery and Resolution from Full-Text Scientific Papers: The PRAISED Framework. *Bio-Algorithms and Med-Systems*, 8.

Toti, D. and Rinelli, M. (2016). On the road to speed-reading and fast learning with CONCEPTUM. In *Proceedings - 2016 International Conference on Intelligent Networking and Collaborative Systems, IEEE INCoS 2016*, pages 357–361.

Vidhya, K. and Aghila, G. (2010). A survey of naive bayes machine learning approach in text document classification. *Int. J. of Com. Sc. and Inf. Sec.*, 7:206–211.

Wang, L. and Zhao, X. (2012). Improved k-nn classification algorithm research in text categorization. In *Proc. of the 2nd Int. Conf. on Comm. and Net. (CECNet)*, pages 1848–1852.

Wang, S. and Manning, C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proc. of the 50th Annual Meeting of the ACL: Sh. P. Vol. 2*, pages 90–94. ACL.

Wetzker, R. and et al. (2008). Tailoring taxonomies for efficient text categorization and expert finding. 3:459–462.