The Impact of Clustering for Learning Semantic Categories

Mário Antunes[1], Diogo Gomes[1,2] and Rui L. Aguiar[1,2]
[1]*Instituto de Telecomunicações, Universidade de Aveiro, Aveiro, Portugal*
[2]*DETI, Universidade de Aveiro, Aveiro, Portugal*

Keywords: Clustering, Text Mining, Machine Learning, IoT, M2M, Context Awareness.

Abstract: The evergrowing number of small devices with sensing capabilities produce massive amounts of diverse data. However, it becomes increasingly difficult to manage all these new data sources. Currently, there is no single way to represent, share, and understand IoT data, leading to information silos that hinder the realization of complex IoT/M2M scenarios. IoT/M2M scenarios will only achieve their full potential when the devices become intelligent: communicate, work and learn together with minimal human intervention. Pursuing these goals, we developed methods to estimate semantic similarity based on distributional profiles. Cluster algorithms were used to learn semantic categories and improve the model accuracy. In this paper, we discuss the impact of the clustering algorithm and respective heuristic to estimate input parameters for the task of learning semantic categories. Our evaluation has shown that *K*-means combined with silhouette methods achieved the higher result.

1 INTRODUCTION

Mining information from data has been the main task of computers over the past years. Yet, the focus is shifting, computers of the near future will be used to extract knowledge from information. This shift can be explained by the ever-growing number of devices, a direct consequence of the Internet of Things (IoT) (Wortmann et al., 2015) and machine-to-machine (M2M) (Chen and Lien, 2014).

The sheer volume and complexity of data make it difficult to extract possible relevant information and knowledge. Data gathered by these devices has no value in its raw state, it must be analysed, interpreted and understood. Context-awareness computing plays an important role in tackling this issue (Perera et al., 2014), and is an intrinsic property of IoT and M2M scenarios.

In IoT and M2M scenarios, its possible to learn/ detect an entity's context by combining data from multiple sources. An entity's context can them be used to provide added value: improve efficiency, optimize resources and detect anomalies. However, recent projects still follow a vertical approach (Fantacci et al., 2014; Robert et al., 2016; Datta et al., 2016). Devices/manufacturers do not share context information, or share it with a different structure, leading to low interoperability and information silos respectively. This has hindered interoperability and the realization of even more powerful IoT scenarios, which are being pursued mostly by large corporations with a dominant market position and considerable resources (*e.g.* Google, Amazon and Microsoft). Another important issue is the need felt for a new way to manage, store and process such diverse machine made context information; unconstrained and without limiting structures. The full potential of IoT and M2M scenarios will only be achieved when we overcome these limitations.

In previous publications, we addressed the abovementioned issues and developed an organization model that relies on machine learning features to organize context information. At the time we explored two types of features: semantic (Antunes et al., 2017a) and stream similarity (Jesus et al., 2017).

Regarding semantic similarity, we developed a model named Distributional Profile of Word Categories (*DPWC*) and an unsupervised method to learn the model from common web services, such as search engines. The learning method uses clustering algorithms to identify word categories automatically (word category is closely related to concepts and the possible meaning of a word). Our prototype relied on *k*-means to cluster the distributional profile into categories. However, the algorithm has two main disadvantages: is not deterministic and requires the num-

Antunes, M., Gomes, D. and Aguiar, R.
The Impact of Clustering for Learning Semantic Categories.
DOI: 10.5220/0006813603200327
In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018)*, pages 320-327
ISBN: 978-989-758-296-7

ber of clusters *a priori*. In this paper, we explore other clustering methods and study their impact on category learning. Since the *DPWC* learning method is unsupervised, we are interested in automatic methods to estimate the clustering parameters.

The remainder of the paper is organized as follows. In Section 2 we detail our context organization model. Section 3 summarizes the *DPWC* semantic model. An overview of clustering algorithms is given in Section 4. In Section 5 we discuss the clustering algorithms and parameters' estimation heuristic used. Section 6 gives important details about our prototype. The results of our evaluation are given in Section 7. Finally, discussion and conclusions are presented in Section 8.

2 CONTEXT ORGANIZATION MODEL

Context information is an enabler for further data analysis, potentially exploring the integration of an increasing number of information sources. Common definitions of context information (Abowd et al., 1999; Dey, 2001) do not provide any insight about its structure. In fact, each device can share context information with a different structure. One important objective of context representation is to standardize the process of sharing and understanding context information. However, nowadays no widely accepted context representation scheme exists; instead, there are several approaches to deal with context information. These can be divided into three categories: i) adopt/ create a new context representation, ii) normalize the storing process through ontologies, iii) accept the diversity of context representations.

We accepted the diversity of context representation as a consequence of economic pressures and devised a bottom-up model (Antunes et al., 2016) to organize context information without enforcing a specific representation. Our organization model is divided into four main parts, as depicted in Figure 1. The first two components represent the structured part of our model and account for the source ID and fixed *d*-dimensions respectively. These *d*-dimensions allow human users to select information based on time, location or even other dimensions and can be understood as an OLAP cube helping in the process of filtering information. The remaining components of our model extract information from the content itself and organize it based on semantic and stream similarity. This paper focuses on semantic features, especially the impact of clustering on category detection.

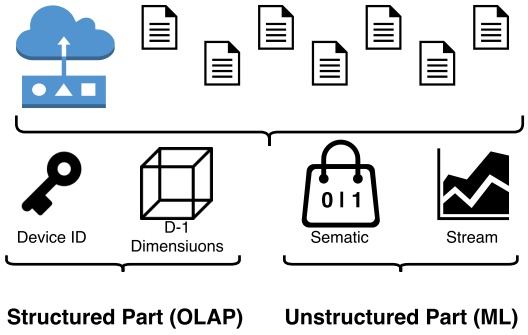

Figure 1: Context organization model based on semantic and stream similarity.

3 DISTRIBUTIONAL PROFILE OF WORD CATEGORIES

Semantic distance/similarity is a property of lexical units, typically between words but this notion can be generalized to larger units such as phrases, sentences, etc. Two words are considered semantically close if there is a lexical semantic relation between them. There are two types of lexical relations: classical relation (such as synonyms, antonyms and hypernymy) and ad-hoc non-classical relation (such as cause-and-effect). If the closeness in meaning is due to a certain classical relation, then the terms are said to be semantically **similar**. On the other hand, semantic **relatedness** is the term used to describe the more general form of semantically closeness caused by any semantic relation. For instance the nouns *liquid* and *water* are both semantically similar and related, whereas the nouns *water* and *boat* are semantically related but not similar.

Semantic features allow us to estimate similarity between concepts. This similarity allows us to organize, extract and cluster information based on concepts and not on sub-strings nor regular expressions. In other words, the devices are able to autonomously learn concepts and not only strings. These concepts provide latent knowledge to the underlying information and do not depend on human users or context representation. This is especially important for IoT/ M2M scenarios. IoT/M2M devices share a diverse amount of data.

Given a target word u we use public web services, namely search engines, to gather a potentially relevant corpus and extract the word u distributional profile. The profile is built based on proximity, which means if a word w is within the neighbourhood of a target word u it is properly processed and extracted. This distributional profile of a word (*DPW*) is defined as

$$DPW(u) = \{w_1, f(u,w_1); ...; w_n, f(u,w_n)\} \quad (1)$$

where u is the target word, w_i are words that occur within the neighbourhood of u and f stands for co-occurrence frequency (can be generalized for any strength of association metric). A distributional profile can also be interpreted as a vector that represents a point in high dimensional space, each word w_i represent a dimension and $f(u,w_i)$ represents its value in that dimension. From this point onward we will refer to words inside a *DPW* as dimensions. We evaluate the similarity between two *DPW* with cosine similarity:

$$S(u,v) = \frac{\sum_{i=1}^{n} f(u,w_i) \times f(v,w_i)}{\sqrt{\sum_{i=1}^{n} f(u,w_i)^2} \times \sqrt{\sum_{i=1}^{n} f(v,w_i)^2}} \quad (2)$$

Other similarity measures can be used, however cosine is invariant to scale, which means it does not take into account the vector's magnitude, only their direction. This property is important for unbalanced corpus, such as corpus in M2M scenarios or corpus gathered from web services (due to the ranking algorithms used by web-services).

Although public web services offer some important advantages, they also have some disadvantages. Distributional profiles can be noisy and contain several dimensions with low relevance. A dimension with low relevance is a dimension with a low value of co-occurrence frequency ($f(u,w_n)$). The combined weight of several low relevance dimensions can change the direction of the word vector and damage the cosine similarity. Also, a profile can contain several senses of the target word (sense-conflation). Multiple words senses in a single profile may also change the word vector direction and decrease accuracy, limiting the potential of this method.

In order to minimize the above-mentioned issues, we propose using clustering (overview of clustering methods in Section 4) on the distributional profile to identify categories/word senses. The rationale is that dimensions belonging to the same category are closer to each other than words from other categories. Clustering methods require a distance metric in order to group similar elements. Co-occurrence was used as a distance metric to cluster the dimensions into categories.

It is important to mention that the clusters do not represent word senses from a Roget-style thesaurus (Jarmasz and Szpakowicz, 2004). Which means that there is not a one-to-one relation between the clusters and a word in a thesaurus. Conceptually the clusters are more similar to categories in latent semantic analysis, and may not have a correspondence to our human perception. Since a cluster may not represent a classical word sense, from this point onward we will refer to them as categories. One implication of this statement is that some clusters represent high relevance categories, while others represent low relevance categories. Consider the following scenario, two target words u and v are not related but may end up with the same low relevance category. These categories match with each other and produce a false positive.

In order to minimize this issue, our model incorporates an affinity value between the target word and each category (can be understood as a bias) that measures the natural tendency from a word to be used as a specific category. The affinity is computed as the average similarity between the target word and all the cluster's elements. After the clustering and computing the affinity of the target word to each cluster, the distributional profile of multiple words categories (*DPWC*) is extracted from the *DPW* and grouped according to the clusters obtained. After computing all the affinity values, they are normalized between $]0,1]$ with the following expression

$$a'_i = \frac{a_i}{max(a)} \quad (3)$$

The profile is defined as follows:

$$DPWC(u) = \left\{ \begin{matrix} a_1;\ DPW_1(u) \\ \dots \\ a_m;\ DPW_m(u) \end{matrix} \right\} \quad (4)$$

where u is the target word, $DPW_i(u)$ is the distributional profile for the word category i and a_i is the affinity between u and a word category.

Finally, the similarity between two *DPWC* is given by the following expression

$$S(u,v) = max(cosine(u_c, v_c) \times (a_{u_c} + a_{u_v} \big/ 2)) \quad (5)$$

where u_c and v_c represent a specific category from u and v respectively and a represents the category's affinity. Our final similarity measure is the maximum similarity between all the possible categories weighted by the average category's affinity. By incorporating affinities our model minimizes the effect of low relevance categories.

4 BACKGROUND AND RELATED WORK

Machine learning provides methods that automatically learn from data and accurately predict future data based on what we learn from current observations (generalization). In classification tasks, we have a labelled set of training points, and we learn

the grouping structure in a supervised way. In other words, classification tries to predict the label of (unlabelled) data. However, when a labelled set is not available, it is possible to groups data into "natural" categories without class labels. This task is named clustering.

Cluster algorithm deals with finding some structure in a collection of unlabeled data, as such are considered unsupervised learning. A loose definition of clustering could be the process of "organizing objects into groups whose members are similar in some way". A cluster is a collection of objects which are "similar" between them and are "dissimilar"to the objects belonging to other clusters. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Clustering algorithms can be organized into several taxonomies, one of the most common characterizations is by the underlying model. In the following list, we present four classes of clustering algorithms.

1. **Connectivity-based clustering:** Or hierarchical clustering, these models are based on the notion that closer data points exhibit more similarity to each other than the points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm (Ward, 1963) and its variants.
2. **Centroid-based models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. The number of clusters is required beforehand, which makes it important to have prior knowledge about the dataset. *K*-Means clustering algorithm (Lloyd, 1982) is a popular algorithm that falls into this category.
3. **Distribution-based models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (*e.g.* Normal, Gaussian). A popular example of these models is Expectation-maximization algorithm (Dempster et al., 1977) which uses multivariate normal distributions.
4. **Density-based Models:** These models search the data space for areas with high density of data points. It isolates them into different density regions and assign the data points within these regions to the same cluster. Popular examples of density models are DBSCAN(Ester et al., 1996) and OPTICS(Ankerst et al., 1999).

In Section 5 we discuss the algorithms considered and the parameters' estimation heuristics evaluated.

5 LEARN SEMANTIC CATEGORIES THROUGH CLUSTERING

As previously stated, we use clustering algorithms to learn semantic categories from distributional profiles. This minimizes the issue of sense-conflation and improves the semantic similarity estimation. However, most clustering algorithms require fine-tuning in order to achieve good results.

In this paper, we discuss the impact of clustering algorithm and the possible methods to estimate the initial parameters on the task of learning categories. In our previous work (Antunes et al., 2017a) we used *K*-means++ (Arthur and Vassilvitskii, 2007) with a method for estimating the number of clusters similar to gap statistics (Tibshirani et al., 2001), without having to generate reference features based on the elements to compare the clustering with a uniform sample. The framework used was proposed in (Pham et al., 2005).

Although this combination provided good results, it is unstable. Since *k*-means algorithm is not deterministic, each time it is executed can lead to different results. One way to deal with this issue is to run the algorithm several times and only consider the set of clusters with lower distortion, the obvious drawback is the performance penalty.

With this in mind, we intended to explore alternative clustering algorithms. The ideal cluster algorithm should be reasonably fast (if possible deterministic) and require little no none initial parameters. Hierarchical-based and Distribution-based clustering are not ideal techniques for this task. Hierarchical-based clustering is computationally expensive and requires a method to extract cluster from the hierarchy. *DPW* models are highly dimensional by nature, meaning that parameter estimation is rather difficult. Distribution-based clustering requires a distribution model to fit the data points. As mentioned, the high dimensionality associated with *DPW* makes it difficult to fit a distribution. Taking this into account we selected for this evaluation three algorithms: *K*-means++, DBSCAN and Fast Greedy Clustering (FGC).

K-means++ only requires the number of clusters *a priori*. Several algorithms can be used to estimate this parameter, we evaluated the following approaches. **Elbow method**, this method computes the distortion for each set of clusters, it takes into account the percentage of variance explained as a function of the number of clusters: one should choose a number of clusters so that adding another cluster doesn't give much better modelling of the data. More precisely, if one plots the average distortion against the number of clusters. The first clusters will add much information, but at some point the marginal gain will drop, giving an angle in the graph (named "elbow"). We used Kneedle algorithm (Satopaa et al., 2011) to identify the elbow. **Average silhouette method**, this approach measures the quality of a clustering, it determines how well each object lies within its cluster. It uses a metric named silhouette (Rousseeuw, 1987), a high average silhouette width indicates a good clustering. Average silhouette method computes the average silhouette for different values of *k*. The optimal number of clusters *k* is the one that maximizes the average silhouette. Finally, **gap statistics** can be used to estimate the ideal number of clusters, as mentioned we used an alternative framework to achieve the same result.

DBSCAN clusters data points based on point density, one important advantage is that it can produce cluster with any shape. It requires two parameters, ε and minPts. ε is the maximum distance it uses to search for neighbour data points, and minPts is the minimal number of neighbours to be considered a dense region. Unfortunately, to the best of our knowledge, there is no method or heuristic to identify these two parameters. As a rule of thumb, a minimum minPts can be derived from the number of dimensions D in the dataset, however, the nature of *DPW* makes it difficult to select the ideal minPts. Although distributional profiles are highly dimensional, there are few data points. In fact, each data point is a dimension, the number of data points is equal to the number of dimensions. However, it is possible to select the ideal ε based on the number of minPts. The value for ε can then be chosen by using a *k*-distance graph, plotting the average distance to the *k* nearest neighbours ordered from the largest to the smallest value (Schubert et al., 2017). Good values of ε are where this plot shows an "elbow", again we used the Kneedle algorithm to automatically identify the "elbow". In our evaluation, we tested several values of minPts (from 3 to 5, more than 5 isn't relevant since almost no data point had 6 neighbours), and returned the set of clusters with lower distortion.

FGC is a clustering algorithm developed for a specific scenario (Antunes et al., 2017b), cluster GPS data points to identify potholes. Algorithm 1 describes in detail the inner-workings of the method. It is a cross between DBSCAN and *K*-means, iterates over the points a single time to identify potential pairs but only merges them if the resulting cluster does not invalidate the radius parameters. It has two advantages, it automatically identifies the number of clusters and only requires a single parameter, a maximum radius for the clusters. This parameter controls the growth of each cluster. We evaluated two different heuristics to estimate the radius. The first one was similar to the DBSCAN method, fixed a number of neighbours and selected the ideal elbow from a *k*-distance graph. Tested several values of neighbours, and selected the ones that generate a cluster with smaller distortion. The second one estimated the radius based on threshold algorithms, these types of algorithm separate data points in two different groups (high and low groups). These methods are used in image processing to convert a grey scale image into black and white. The algorithm used to estimate the ideal threshold was IsoData (Ridler and Calvard, 1978), due to its simplicity this algorithm can be adapted to several use cases.

Algorithm 1: Fast Greedy Clustering Algorithm.

```
1:  function CLUSTERING(points, radius)
2:      kdTree ← KDTree.init(points, radius)
3:      pairs ← {}
4:      for each p ∈ points do
5:          neighbours ← kdTree.near(p, radius)
6:          pairs ← pairs + neighbours
7:      end for
8:      clusters ← {}
9:      for each p ∈ pairs do
10:         c ← findClosestCluster(clusters, p)
11:         if maxRadius(c + p) < radius then
12:             c ← c + p
13:         end if
14:     end for
15:     return clusters
16: end function
```

6 IMPLEMENTATION

In this section, we discuss some relevant details about our prototype implementation. Our prototype is divided into 5 different components as depicted in Figure 2. All the components were written in Java.

The first component (corpus extraction) bridges our solution with web search engines. Given a target

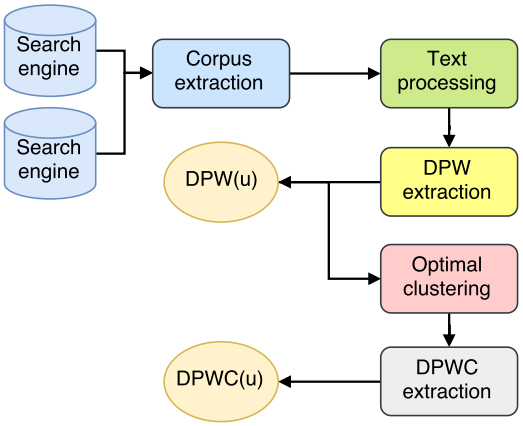

Figure 2: Proposed DP extraction system's architecture.

word u our prototype uses web search engines to extract its $DPW(u)$ and $DPWC(u)$. It can be used with any search engine, and currently it uses three: Faroo[1], Yacy[2]and Searx[3]. This component basic function is to extract a corpus from search engines. The corpus is composed of snippets returned by searching for the target word. In a previous work (Antunes et al., 2017a) we compared the impact of using only snippets against the full web-pages. We observed that snippets contain enough information to build reliable *DPW*s.

The second component (text processing) implements a preprocessing pipeline that cleans the corpus and divides it into tokens. The various spaces of the pipeline are depicted in Figure 3. First the snippets are tokenized and the resulting tokens are filtered using a stop word filter. Stop words are deemed irrelevant because they occur frequently in the language and provide little information. We used the MySQL stop word list[4]. For the exact same reason, we also remove tokens that are too big or too small: any token with less than 3 or more than 14 (9 being the average word length in English) characters was removed from the pipeline.

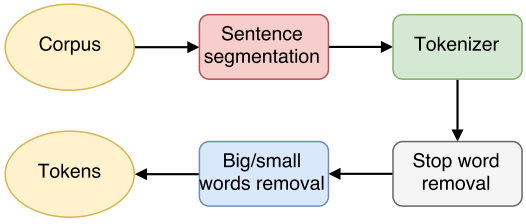

Figure 3: Text processing pipeline.

[1]http://www.faroo.com/hp/api/api.html
[2]http://yacy.net/en/index.html
[3]https://searx.me/
[4]https://dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html

The *DPW* extraction component analyses the output of the pipeline and extracts the *DPW* of the target word u. After extracting the *DPW*, we cluster the profile dimensions based on co-occurrence similarity. The methods discussed in Section 5 were implemented and used at turns for the evaluation. Finally, the *DPWC* component uses the *DPW* and the clusters to return the $DPWC(u)$ of the target word, this component also computes the affinity between the target word and each category.

7 PERFORMANCE EVALUATION

We evaluate our model against Miller-Charles dataset (Miller and Charles, 1991), the reference dataset for semantic similarity evaluation. It is composed of 30 word-pairs rated by a group of 38 human subjects. The word pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy).

To the best of our knowledge, there is no semantic dataset specifically for IoT/M2M available. In order to evaluate our semantic features against IoT vocabulary, we devised one. We mined a popular IoT plaform[5] to extract the most commonly used terms (ranked by term frequency). The 20 most used terms were collected and organized into 30 word pairs. Each pair was rated on a scale from 0 to 4 by five fellow researchers. Although not as comprehensive as the Miller-Charles dataset, our still reach 0.8 correlation amongst human classification. In a future work, we intend to further explore and improve our dataset. The final similarity of each pair is the average of the previously stated rates. This dataset is publicly available[6] and can be used by other researchers.

Correlation between sets of data is a measure of how well they are related. The correlation r can range from -1 to 1. A r of -1 indicates a perfect negative linear relationship between variables, an r of 0 indicates no linear relationship between variables, finally and an r of 1 indicates a perfect positive linear relationship between variables. In short, the highest correlation indicates the most accurate solution.

We evaluated the performance of $DPW(u)$ and $DPWC(u)$ (using multiple clustering algorithms). Our models were learned from a corpus formed from the top 300 snippets returned by three search engines: Faroo, Yacy and Searx.

The results of the evaluation using Miller-Charles and IoT dataset are listed in Table 1 and Table 2 respectively. The optimal neighbourhood's size for

[5]ThingSpeak: https://thingspeak.com/
[6]https://github.com/mariolpantunes/ml/blob/master/src/main/resources/en-iot-30.csv

Miller-Charles and IoT dataset appears to be 7 and 5 respectively. *K*-means with silhouette method outperforms the other clustering algorithms. In our tests, it proved to be more stable and robust than the statistic framework. The statistic framework appears to be less robust than the non-deterministic behaviour of the clustering algorithm. Through manual verification, we verified that when the clusters better represent semantic categories all three methods preformed well. However, the overall quality of the clusters decreased when elbow and statistic methods have difficulty in identifying the correct number of clusters. Both DBSCAN and FGC generate a single cluster with all the data points. As mentioned, the high dimensionality associated with *DPW* profiles leads to a classical curse of dimensionality problem. Since each data point is also a dimension, it becomes a highly dimensional clustering problem with few examples. In short, the heuristic used to estimate the optimal distances (either ε or radius) selected a distance large enough to group all the data points into a single cluster.

In a future publication, we will address this issue and devise new methods to correctly estimate the distance for DBSCAN and FGC for this specific scenario. In the mean time, *k*-means combined with average silhouette method appears to be the ideal method to learn semantic categories from distributional profiles.

Table 1: Performance evaluation on Miller-Charles dataset.

Methods	Neighborhood size 3	5	7
DPW	0.32	0.37	0.45
DPWC (*k*-means elbow)	**0.37**	0.27	0.25
DPWC (*k*-means silhouette)	0.34	**0.45**	**0.61**
DPWC (*k*-means statistic)	0.28	0.41	0.46
DPWC (DBSCAN elbow)	0.32	0.37	0.45
DPWC (FGC elbow)	0.32	0.37	0.45
DPWC (FGC IsoData)	0.32	0.37	0.45

Table 2: Performance evaluation on IoT dataset.

Methods	Neighborhood size 3	5	7
DPW	0.25	0.38	0.32
DPWC (*k*-means elbow)	0.35	0.32	0.39
DPWC (*k*-means silhouette)	**0.37**	**0.51**	**0.39**
DPWC (*k*-means statistic)	0.15	0.19	0.29
DPWC (DBSCAN elbow)	0.25	0.38	0.32
DPWC (FGC elbow)	0.25	0.38	0.32
DPWC (FGC IsoData)	0.25	0.38	0.32

8 CONCLUSIONS

The number of IoT devices is increasing at a steady step. Each one of them capable of generating massive amounts of diverse data. However, each device/ manufactures share context information with a different structure, hindering interoperability in IoT and M2M scenarios.

In a previous publication, we developed a model named Distributional Profile of Word Categories (*DPWC*) and an unsupervised method to learn the model from common web services, such as search engines. The learning method uses clustering algorithms to identify word categories automatically (word category is closely related to concepts and the possible meaning of a word). The implementation relied on *k*-means to cluster the distributional profile into categories. However, there are several clustering algorithms, each one of them with specific advantages. It is important to mention the high dimensionality associated with *DPW* distributional profiles. In this paper, we explore the impact of clustering on category detection. Since the *DPWC* learning method is unsupervised, we are also interested in automatic methods to estimate the clustering parameters.

For our evaluation, we selected *k*-means, DBSCAN and FGC as good candidates to learn semantic categories. Each of them has at least a heuristic to estimate the ideal parameters. All of them were evaluated using two different datasets Miller-Charles dataset (Miller and Charles, 1991) and an IoT semantic dataset. Based on our results we can state that *k*-means combined with silhouette method appears to be the ideal combination to learn semantic categories. The other two clustering algorithms appear to suffer from the curse of dimensionality and outputted a single cluster with all the data points. This implies that the heuristics used were not able to capture the relevant characteristics.

There is still room for improvement, hypernyms can be used to learn more abstract dimensions improving performance and decrease the size of each distributional profile. Non-negative matrix factorization can also be used to discover latent semantic information in distributional profiles and increase accuracy (by estimating the value of dimensions with value zero). Furthermore, other elbow detection method can be devised and tested. This algorithm has a crucial role is selecting the ideal ε and radius for DBSCAN and FGC respectively. We intend to explore several of the previous mentions optimizations and improve our model. Nevertheless, our model was able to learn distributional profiles from a small corpus, achieving a relatively high accuracy on both datasets.

ACKNOWLEDGEMENTS

The present study was developed in the scope of the **Smart Green Homes** Project [POCI-01-0247-FEDER-007678], a co-promotion between **Bosch Termotecnologia S.A.** and the **University of Aveiro**. It is financed by Portugal 2020 under the Competitiveness and Internationalization Operational Program, and by the European Regional Development Fund. This work was also partially supported by research grant SFRH/BD/94270/2013.

REFERENCES

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Proc. of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). OPTICS: ordering points to identify the clustering structure. *ACM SIGMOD Record*, 28(2):49–60.

Antunes, M., Gomes, D., and Aguiar, R. L. (2016). Scalable semantic aware context storage. *Future Generation Computer Systems*, 56:675–683.

Antunes, M., Gomes, D., and Aguiar, R. L. (2017a). Towards IoT data classification through semantic features. *Future Generation Computer Systems*.

Antunes, M., Gomes, D., Barraca, J. P., and Aguiar, R. L. (2017b). Vehicular dataset for road assessment conditions. In *Procedings in the third IEEE Annual International Smart Cities Conference (ISC2 2017)*.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035. Society for Industrial and Applied Mathematics.

Chen, K.-C. and Lien, S.-Y. (2014). Machine-to-machine communications: Technologies and challenges. *Ad Hoc Networks*, 18:3–23.

Datta, S. K., Bonnet, C., Costa, R. P. F. D., and Härri, J. (2016). Datatweet: An architecture enabling data-centric iot services. In *2016 IEEE Region 10 Symposium (TENSYMP)*, pages 343–348.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Fantacci, R., Pecorella, T., Viti, R., and Carlini, C. (2014). Short paper: Overcoming iot fragmentation through standard gateway architecture. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 181–182.

Jarmasz, M. and Szpakowicz, S. (2004). Roget's thesaurus and semantic similarity. In *Recent Advances in Natural Language Processing III*, page 111. John Benjamins Publishing Company.

Jesus, R., Antunes, M., Gomes, D., and Aguiar, R. (2017). Extracting knowledge from stream behavioural patterns. In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*. SCITEPRESS - Science and Technology Publications.

Lloyd, S. (1982). Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, 16(1):414–454.

Pham, D. T., Dimov, S. S., and Nguyen, C. D. (2005). Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119.

Ridler, T. and Calvard, S. (1978). Picture thresholding using an iterative selection method. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(8):630–632.

Robert, J., Kubler, S., Traon, Y. L., and Främling, K. (2016). O-mi/o-df standards as interoperability enablers for industrial internet: A performance analysis. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4908–4915.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Satopaa, V., Albrecht, J., Irwin, D., and Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). DBSCAN revisited, revisited. *ACM Transactions on Database Systems*, 42(3):1–21.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Wortmann, F., Flüchter, K., et al. (2015). Internet of things. *Business & Information Systems Engineering*, 57(3):221–224.