# SEPARAÇÃO AUTOMÁTICA DE ATRIBUTOS PARA MÉTODOS DE APRENDIZADO MULTI-VISÃO

LUIZ FELIPE GONÇALVES MAGALHÃES

# SEPARAÇÃO AUTOMÁTICA DE ATRIBUTOS PARA MÉTODOS DE APRENDIZADO MULTI-VISÃO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais — Departamento de Ciência da Computação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Marcos André Gonçalves
Coorientador: Daniel Hasan Dalip

Belo Horizonte

Fevereiro de 2018

Magalhães, Luiz Felipe Gonçalves

M188s Separação automática de atributos para métodos de aprendizado multi-visão / Luiz Felipe Gonçalves Magalhães. — Belo Horizonte, 2018
xii, 54 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais — Departamento de Ciência da Computação
Orientador: Marcos André Gonçalves
Coorientador: Daniel Hasan Dalip

1. Computação — Teses. 2. Recuperação da Informação. 3. Aprendizado do Computador. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*73(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

SEPARAÇÃO AUTOMÁTICA DE ATRIBUTOS PARA MÉTODOS DE APRENDIZADO MULTI-VISÃO

**LUIZ FELIPE GONÇALVES MAGALHÃES**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Marcos André Gonçalves
Prof. Marcos André Gonçalves - Orientador
Departamento de Ciência da Computação - UFMG

Daniel Hasan Dalip
Prof. Daniel Hasan Dalip - Coorientador
Departamento de Computação - CEFET/MG

[illegible]
Profa. Gisele Lobo Pappa
Departamento de Ciência da Computação - UFMG

[illegible]
Prof. Mario Sérgio Ferreira Alvim Júnior
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 4 de maio de 2018.

# Resumo

Aprendizado Multivisão é uma tendência em alta em aprendizado de máquina e já produziu resultados bastante significativos em diversas áreas de aplicação. Uma delas é a de verificação automática de qualidade de conteúdo criado colaborativamente na Web, melhor exemplificado pelas 'Wikis'. Wikis são uma das formas mais comuns de repositórios de informação, os quais usuários utilizam quando se encontram sobre alguma necessidade de informação. Dada sua natureza livre e colaborativa, tais repositórios precisam de uma forma de controlar a qualidade de seu conteúdo, com objetivo de evitar incluir informações erradas e/ou incompletas. A solução estado-da-arte para esse problema se apoia no aprendizado multivisão, em que qualidade é considerada um conceito multifacetado que pode ser aprendido partindo de critérios de qualidade definidos por humanos. Para esse efeito, atributos que descrevem qualidade devem ser criados e agrupados em visões baseadas em critérios como a estrutura do texto, legibilidade, estilo, histórico de edição do usuário, entre outros. A tarefa de determinar as visões requer a assistência de um especialista, o que é dificil de executar em cenários onde visões se sobrepõem ou são difíceis de serem interpretadas por humanos. Além disso, visões definidas manualmente podem não ser as mais adequadas para resolver automaticamente o problema de verificação de qualidade. Nesse trabalho é proposto um gerador de visões automático, para endereçar o problema da geração de visões para aprendizado multivisão, especialmente para o problema de verificação automática de qualidade. Foram avaliados três conjuntos de dados Wiki bastante populares utilizando a abordagem proposta. Os experimentos realizados mostraram que a solução proposta teve performace mais alta que uma versão que utiliza apenas os atributos originais, com ganhos chegando a até 20% em termos de acurácia da qualidade prevista. O método desenvolvido nesse trabalho foi também capaz de automaticamente produzir visões que são competitivas ou até melhores do que as criadas manualmente, para a mesma tarefa de verificação de qualidade, e sem nenhuma interferência humana.

**Palavras-chave:** Aprendizado de Máquina, MultiVisão, Verificação de Qualidade,

Recuperação de informação.

# Abstract

Multi-view learning is a “hot” tendency in machine learning that has produced top-notch results in several application areas. One of them is automated quality assessment of content created collaboratively on the Web, better exemplified by ‘Wikis’. Wikis are one of the most common information repositories, to which users resort when they have some information need. Given their free and collaborative nature, such repositories need to control content quality, in order to avoid containing wrong or incomplete information. The state-of-the-art solution for this problem relies on multi-view learning, where quality is considered a multifaceted concept that can be learned from human quality assessments. To this effect, features describing quality have to be devised and grouped into views based on criteria such as text structure, readability, style, user edit history, etc. The task of determining the views requires the assistance of an expert, which is hard to do in scenarios where views are overlapping or hard to interpret by humans. In addition, human engineered views may not be the most adequate for automatically solving the quality measurement problem. In this work, we propose an automatic view generator, to address the problem of generating views for MultiView learning, specially for the problem of automated quality assessment. We evaluate this approach on three popular Wiki datasets. In our experiments, our solution outperformed a version that exploits only the original features, with gains of up to 20% in terms of accuracy of the quality assessment. Our method was also able to automatically produce views that are competitive or even better than those manually created, for the task of quality assessment, without any human intervention.

**Keywords:** Machine Learning, Multiview, Quality Assessment, Information Retrieval.

# Lista de Figuras

# Lista de Tabelas

# Sumário

# Capítulo 1

# Introduction

## 1.1 Motivation

Many tasks in digital libraries, such as topic classification, document quality assessment, or link prediction, have been addressed using machine learning approaches [Dalip et al., 2014, 2017; Ferreira et al., 2015]. In such approaches, diverse sets of evidence can be explored during the learning process. For instance, to classify the polarity of an opinion expressed in a video, patterns can be extracted from different sets of evidence, such as the contents of the audio and image streams. Even for a particular source, different sets of features can be observed. For example, the audio source can be described by the spoken content or the emotional voice stress; each of these sets of features can be also classified according to their representation domains, such as time- and frequency-based.

Such scenarios, where data can be described by different sets of features, known as *views*, are increasingly common [Sun, 2013].

Each view corresponds to a partition of the set of features where the features of a particular view are naturally seen as a group. Further, defined as partitions, the views cannot share features which implies that views are designed to be as much independent as possible to each other.

Many works in the literature have addressed the problem of learning from data represented by multiple distinct feature sets. Most of them exploit the consistency and complementary properties of different views to integrate them and provide effective methods, capable of better generalization than single-view learning.

Most multi-view approaches in the literature take advantage of views that represent natural splits of the set of features [Dalip et al., 2012, 2013, 2017; Kleminski et al., 2017; Zhao et al., 2017]. Although in some cases these splits are obvious (for instance, web content can be described by linkage and textual features), in many scenarios, a human expert would be required to determine how the set of features should be partitioned to compose the views. As a result, (a) these methods are not fully automated since they depend on human expertise; (b) they cannot be applied in scenarios where the features are not easily understandable by humans; and (c) solutions proposed by humans, even if experts, may be sub-optimal when compared to pure data-driven techniques.

To address these problems, some approaches estimate the feature splits by means of data analysis. We refer to these methods as data-driven split methods. Their general idea is to partition the feature set into views according to how the features are correlated to each other. Correlations are estimated with respect to a target variable, such as labels provided in classification tasks. As with natural views, improvements in performance have been observed for data-driven split methods [Sun, 2013; Dang & Croft, 2010; Sun et al., 2011; Di & Crawford, 2012; Kleminski et al., 2017]. Nevertheless, data-driven split methods still have some shortcomings, such as the assumption of a hard view consistency restriction and the loss of information associated with individual features, given that they are replaced by the generated views.

The view consistency restriction establishes that a feature belongs to a single view [Gao & Yang, 2014]. This can be a problem because some features can be associated with multiple views. For instance, in a data quality classification problem [Dalip et al., 2017], a feature based on a part-of-speech tag can be associated with a writing style view and with a readability view. Also, many views can be dependent on each other.

The loss of information associated with individual features is sometimes observed after they are integrated into a view. In previous works [Dalip et al., 2012, 2013, 2017], we have observed that the substitution of individual features by their corresponding views sometimes breaks important interactions among features from different views. For instance, in a sentiment analysis problem, by combining the audio view with the subtitles view for a negative sentence $s$ in a video, the classifier can label $s$ as positive. However, by combining a particular spoken word (a feature from the subtitle view) with the corresponding voice intonation (a feature from the audio view) the classifier can capture an irony, correctly labelling $s$ as negative.

## 1.2 Objetives

In this work, we propose an automatic data-driven split method to multiview learning to tackle the aforementioned issues. Our approach extends the method proposed by [Dalip et al., 2012, 2013, 2017], by automatically creating views independently on the dataset to be learned. Although our method is domain and dataset independent, in here we focus on the problem of automated quality assessment of collaborative content created on the Web as in [Dalip et al., 2012, 2017], better exemplified by 'Wiki-like' sites. We chose such domain since it naturally induces a considerable number of views based on different quality criteria such as text structure and style, readability, user review history, etc.

As in previous approaches [Dalip et al., 2012; Di & Crawford, 2012; Gao & Yang, 2014], we split the feature set based on feature correlations, such that similar features are combined into views to capture local consistency. However our proposal differs from previous works in the following important aspects:

1. The adoption of *Soft* views. Instead of taking views as hard partitions of the feature set, we group features such that they can be shared among the views. This way, we relax the consistency rule restriction in an attempt to better capture natural view dependencies.

2. The search for views composed by dissimilar features to create views that also capture interactions among non-correlated features.

3. Preservation of the individual features in the final combination. We combine the views with all the individual features to avoid the loss of information associated with the substitution of individual features. Differently from Dalip et al. [2012, 2013, 2017], we provide a comprehensive study on the preservation of individual features.

4. The iterative evaluation of view candidates. Our method uses label information to estimate feature correlations and, by extent, view candidates. This is an iterative process, in which new view candidates are created at each iteration. However, only sets of views which improve evaluation are preserved.

5. Selection of only the most discriminative views. We propose ways of filtering out and removing potential noise or redudancy regarding the automatically created views, improving the overall process.

In this thesis we focus on comparing the performance of our new automated approach to the manual views proposed in Automatic Quality Assessment domain, by using the collaborative content created on the Web as in [Dalip et al., 2012, 2017].

We, then, guide our analysis by exploring on the following research questions:

1. *Q1 - How effective can be an automatic view generation method compared to a natural view split created by an expert?* We propose an automatic view generator method called Iterative K-Medoids View Generator (IKMVG), and then compare its effectiveness against the manual views in WIKI datasets. In most situations we have similar or even better results compared with the natural (i.e manual/view split) approach.

2. *Q2 - What is the impact of relaxing the view consistency rule, and allowing different views to have some equal features?* We propose an automatic view generator method that explores this aspect with an approach called 'soft views'. Our method iteratively generates views from the data by clustering a selected number of features into groups and use them according to their performance (by evaluating the effectiveness of the generated views). While a feature can not exist in two different views during the same iteration of the clustering, views created on next iterations certainly can. Our results showed that this soft approach can have some benefits, even improving upon the manual view split.

3. *Q3 - How impactful is the usage of similar metrics to generate artificial views?* In this work we used the Spearman Correlation Coefficient to address how similar a feature is to the others, and then group them by this metric. We explore similarity and dissimilarity, by creating views with both metrics. The idea behind this is to allow our method to create views with specific information and views with more general information. This showed interesting results, with strong results against the manual approach and even after comparing to the Random Subspace Method (RSM), which generates views with random features on each of them.

4. *Q4 - How similar can be the views automatically generated to the manual ones?* In this work we realize an indepth analysis on how similar the artificial views can be to the manual ones. Given that, initially, the manual approach is supposed to generate the best possible ones. In our experiments we were able to generate some similar views, specially when a manual view is easily differentiable from the others. In this scenario, our method was able to even replicate the 'Readability' view.

## 1.3 Contributions

Given our research questions, the main contributions of this work are:

- We developed a method called Iterative K-Medoids View Generator (IKMVG) to automatically generate views given the data for the application of a multi-view learning approach. It removes the necessity of a manual separation of features into views by a human, and can have similar or even better performance than this separation.
- From what we know, it is the first time an automatic view separation method relaxes the view consistency rule, allowing features to appear in more than one view. Our so called *Soft* views permit more kinds of relationships between features to exist compared to the hard partitioned views approach, which can positively affect the results, given our research.
- We provide an extensive analysis on our automatically generated views compared to the manual ones, to try understand the strong and weak aspects of our method and what we are contributing to each domain evaluated

The following section contains a roadmap with a brief description of the organization of each chapter of this dissertation.

## 1.4 Roadmap

The remaining of this work is organized as follows. In Chapter 2, we provide necessary background information. In Chapter 3, we discuss related work. In Chapter 4, we present a set of manual views used as a baseline in our experiments. In Chapter 5, we provide information about the Multiview Framework and our approach to Multiview generation. In Chapter 6 we present our experimental results and analysis. Finally, in Chapter 7, we present concluding remarks and plans for future work.

# Capítulo 2

# Background

## 2.1 Supervised Learning

Supervised Machine Learning techniques aim at learning, through data patterns, a way to reach some target result with the least amount of error possible [Mitchell, 1997]. To accomplish this we assume we have access to some training data of the form $A = (a_1, r_1), (a_2, r_2), ..., (a_n, r_n)$ where $a_i$ represents one instance of the data described by one or more features and with a target value $r_i$. In this work, the term feature is used to represent a specific measure of the instance it describes, in order to help to predict $r_i$. If $a_i$ is a Wikipedia article, for example, one feature could represent its length. Machine learning methods try to combine all the features to predict a result that is the closest possible to $r_i$. The values of $r_i$ are dependent of the problem we are trying to solve. In some cases $r_i$ can represent finite values, for example a problem of defining if a email is or is not a spam has only two possible values,'true' or 'false'. This is a classic case of a classification problem. In other problems, $r_i$ can represent infinite real numbers and the goal is to predict this number. For example, predict a company's month profit, which is now a problem called regression. In this work we are going to deal exclusively with regression problems, thus the next section explains in more details the method used in this work, called SVR (Support Vector Regression) [Drucker et al., 1997].

## 2.2 Support Vector Regression

Given the training data defined on the previous section $A = (a_1, r_1), (a_2, r_2), ..., (a_n, r_n)$, our problem is to find a function $f$ which approximates the mapping between input

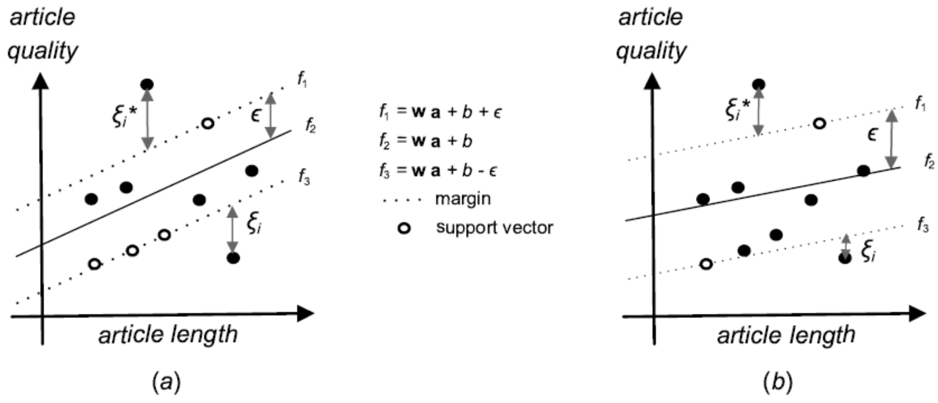

**Figura 2.1.** Regression problem with one numeric target (article quality) and ten articles (points), represented by a single feature (article length). Note that two articles, in both graphics, are considered examples of error because they lie outside the area delimited by the margins. Their distance to the margins are given by $\xi_i^*$ and$\xi_i$ respectively. Figures (a) and (b) represent regressions performed using two different $\epsilon$ values.

and output variables. In our case, the input variables are given by the features used to describe $a_i$, and the output value is our target $r_i$. In this scenario, error for each instance is defined by the difference between the predicted and the true value $(f(a) - r), r \in \mathbb{R}, a \in A_v$. The magnitude of the error is measured by a loss function. The main idea of the SVR is to use a loss function, called $\epsilon - insensitive$, that does not consider error values situated within a certain distance of the true value. One way of visualizing this method is to consider a region of size $\pm\epsilon$ around the hypothesis function $f_2$, where $\epsilon$ denotes a margin. Any training point lying outside this region is considered an example of an error, as illustrated by 2.1(a), where $f_1$ and $f_3$ represent the margins around hypothesis function $f_2$. In this work our goal is to approximate a function $f : A \to \mathbb{R}$ that has at most $\epsilon$ deviation from the output variable $r \in \mathbb{R}$, for all the training data. In SVR, the input $a$ is first mapped onto a $m - dimensional$ feature space using some non-linear mapping $\phi$. A linear model is then constructed in this feature space. More formally, the linear model $f(a, w)$ is given by $f(a, w) = (w, \phi(a)) + b$, where $w$ is the weight vector of $m$ feature values, b is the bias term and $(w, \phi(a))$ denotes the inner product between $w$ and $\phi(a)$. The quality of the estimation is measured by the $\epsilon - insensitive$ loss function $L^\epsilon(r, f(a, w))$ defined below:

$$L^\epsilon(r, f(a, w)) = \begin{cases} |r - f(a, w)| - \epsilon \text{ if } \epsilon < |r - f(a, w)|. \\ 0 \text{ otherwise.} \end{cases} \tag{2.1}$$

SVR basically performs a linear regression in the high-dimension feature space using the $\epsilon - insensitive$ loss function while it tries, at the same time, to reduce the model complexity by minimizing the norm of $w$. The linear regression of the loss function is performed by minimizing error estimates $(r - f(a, w))$ and $(f(a, w) - r)$, measured, respectively, by non-negative slack variables $\xi_i^*$ and $\xi_i$. If we consider $f_1$ the margin above $f$ and $f_3$ the margin below $f$, then $\xi_i^*$ measures deviations above $f_1$ whereas $\xi_i$ measures deviations below $f_3$, as shown in 2.1. Thus, SVR can be formulated as the convex optimization problem of minimizing:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i^* + \xi_i) \tag{2.2}$$

subject to:

$$|r_i - f(a_i, w)| \leq \epsilon + \xi_i^* \tag{2.3}$$

$$|f(a_i, w) - r_i| \leq \epsilon + \xi_i \tag{2.4}$$

$$\xi_i^*, \xi_i > 0, 0 < i \leq n \tag{2.5}$$

where $C > 0$ is a constant parameter. This optimization problem can be transformed into its dual problem, whose solution is given by the equation below:

$$f(a) = \sum_{i=1}^{nSV}(\alpha_i + \alpha_i^*)\kappa(a_i, a), \text{subject to } 0 < \alpha_i, \alpha_i^* \leq C. \tag{2.6}$$

where $n_{SV}$ is the number of support vectors (vectors lying on the margins, depicted as white circles in 2.1) and $\kappa$ is an inner product function (kernel function) defined as $\kappa(a_i, a) = \sum_{j=1}^{m}(\phi_j(a_i)\phi_j(a))$. The SVR estimator accuracy relies on a good setting for $C$, $\epsilon$ and the kernel parameters. C determines the trade-off between model complexity and the degree to which deviations larger than $\epsilon$ are tolerated. If $C$ is large, the objective becomes to minimize the equation $\frac{1}{n}\sum_{i=1}^{n} L^{\epsilon}(r_i, f(a_i, w))$. The  parameter controls the width of the $\epsilon$-insensitive zone, used to fit the training data. Bigger $\epsilon$-values use fewer support vectors, at the expense of providing more 'flat' estimates, as we can see in 2.1(a) and 2.1(b).

In this work we manly selected the SVR algorithm to have a fair comparison with the results presented in previous works [Dalip et al., 2012, 2017], given that we are using the same datasets but proposing a different approach. Another important point is that some of the ascending popularity methods for regression, like the Random Forest approach [Breiman, 2001], do some steps during its tree creation method that can be seen as a partially 'multiview' approach, by selecting only a group of features to belong to each tree. This point would make the analysis of the performance and benefits of our approach a lot more complex. We intend to explore these methods in a future work. In here we used the SVR implementation presented in the SciKit library in Python [Pedregosa et al., 2011]. The parameters $\epsilon$ and $C$ were chosen by a cross-validation within the training set [Mitchell, 1997], while the $\kappa$ parameter selected was $RBF$, the same one used by [Dalip et al., 2012].

## 2.3 Feature Selection

While many factors can impact the sucess of machine learning on a given task, the representation and quality of the data is first and foremost [Mitchell, 1997]. In theory, the increase of the feature space means more information available thus allowing the method to be more discriminative but, in practice, this approach can bring some potential problems. First, the bigger our feature space, the more time consuming our whole learning process will be and it can cause our method to overfit the training data and compromise the generalization of the model. Second, it is possible to find many redundant or noisy features on such big feature space. This information can misguide the learner and increase the amount of non useful information it has to filter. Given these problems, one of the most adopted approaches to solve the issue of a high number of features is the idea of feature selection. A feature selection method has the benefits of facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times and defying the curse of dimensionality to improve prediction performance [Guyon & Elisseeff, 2003].

In this work, as will be explained in subsequent sections, each view generated by our method will become a new feature for the end learner. If a high number of views were to be generated, it is possible that some of them are redundant and would not help in generating a good model. The use of feature selection allow us not to worry so much about generating similar views during our method, given that they will probably be filtered if they do not add any new information.

### 2.3.1 Information Gain

For our experiments we used a feature selection algorithm based on the algorithm called Information Gain [Witten et al., 2016]. The method is used to evaluate the information gained by using each feature while trying to solve a classification problem. To measure the Information Gain, first we need to compute the value of the information, by using the entropy metric. It can be calculated by the following expression:

$$information(p_1, p_2, ..., p_n) = entropy(\frac{p_1}{m}, \frac{p_2}{m}, ..., \frac{p_n}{m}) \tag{2.7}$$

$$entropy(q_1, q_2, ..., q_n) = -q_1 * log_2(q_1) - q_2 * log_2(q_2) - ... - q_n * log_2(q_n) \tag{2.8}$$

where $p_i$ represents the number of items from class $i$ and $m$ is the total number of items. The bigger the entropy, the higher is the disorder level, which means less information.

As an example, based on the values presented in table 2.1 we can calculate the Information Gain of feature A. The first step is to calculate the total information of the system before using A:

$$information(4, 3) = entropy(\frac{4}{7}, \frac{3}{7}) \tag{2.9}$$

$$entropy(\frac{4}{7}, \frac{3}{7}) = -\frac{4}{7} * log_2(\frac{4}{7}) - \frac{3}{7} * log_2(\frac{3}{7}) = 0,985 \tag{2.10}$$

To measure the information gained by using the feature A, we must compute the information when $A = 0$, $A = 1$ and $A = 2$, giving us information(1,1) = 1, information(2,0) = 0 and information(2,1) = 0,918. Finally we can compute the total information from feature A, and then the information gain obtained from feature A by subtracting the information after using A from the information before using A.

$$information_A = \frac{2}{7} * 1 + \frac{2}{7} * 0 + \frac{3}{7} * 0,918 = 0,679 \tag{2.11}$$

$$infoGain_A = information(4, 3) - information_A \tag{2.12}$$

$$infoGain_A = 0,985 - 0,679 = 0,306 \tag{2.13}$$

## 2.4 Spearman Correlation Coefficient

The Spearman Correlation Coefficient measures the strength and the direction between two ranked variables. It is a similar metric to the Pearson Coefficient [Myers et al., 2010], but while Pearson only assesses linear relationships, Spearman's correlation assesses monotonic relationships. In this work the ranked variables are the features, and we use this coefficient to understand how correlated a pair is. Given the features $x$ and $y$, the first step is to produce a ranking for each of them, from the smallest value

**Tabela 2.1.** Generic values of Features A and B, both with 3 possible values, for 7 different items with 2 possible classes

| Item Id | Feature A | Feature B | Class |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 |
| 4 | 1 | 2 | 1 |
| 5 | 2 | 1 | 0 |
| 6 | 2 | 0 | 0 |
| 7 | 2 | 1 | 1 |

(ranked as 1) to the biggest one (ranked as $n$). Using the ranking scores, the coefficient can be calculated using the formula

$$\rho = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2 \sum_i (Y_i - \bar{Y})^2}}. \tag{2.14}$$

Where the index $i$ represents the ranking value of a given feature in instance $i$ of the training data. The correlation between the features $x$ and $y$ has a result between $-1$ and 1, where values next to 1 mean that they have a similar rank and values next to $-1$ represent a dissimilar rank between these two variables.

## 2.5 Unsupervised Learning

Unsupervised learning is the machine learning task that aims to infer a function to describe unlabeled data. It means that, differently from Supervised Learning, there is no access to the target values in the data (used for training in supervised scenarios). Given this situation, unsupervised methods focus more on the idea of creating clusters, or 'natural groupings' of the input patters [Duda et al., 2000]. The idea of views, one of the main aspects adressed in this work, can be seen as clusters, in which features that have some type of similarity are grouped together. This means that clustering techniques have a high potential to help in our scenario of view generation.

There are many approaches to clustering present in the literature, including Hierarchical Clustering and the K-Means clustering [Duda et al., 2000]. In this work we used a variation of the K-Means algorithm during a step of our method for view generation, called K-Medoids [Kaufman & Rousseeuw, 1987] implemented on a R library package.

### 2.5.1 K-means Clustering

The K-means algorithm was first proposed by [MacQueen et al., 1967], and it is one of the most known and used clustering algorithms to date. It has a lot of variations present in the literature. The traditional k-means starts by selecting $k$ points in our vector space to be the initial means. This process may be done in different ways, by selecting them randomly or by directly choosing points that favor the method. (It is generally better to choose points with very different values).

After the means (most commonly called centroids) have been chosen, the distance between all elements and each mean is calculated, and the element with the smaller distance (and more similar to the mean) is grouped to that centroid. After this, the new centroid is updated taking into account the element added. This process is, then, repeated until all elements are associated to one of the means. It is possible, and common, to change an already associated element in future iterations, as K-means is not a greedy algorithm.

The K-means procedure can be viewed as optimizing a loss function:

$$L = \sum_i ||X_i - \mu_{z_i}||^2 \tag{2.15}$$

where $X$ represents the set of points and $\mu_{z_i}$ the centroid (or cluster) were the point $i$ was placed.

For this work we did not use this traditional K-means approach, but one of its variations, called K-Medoid.

### 2.5.2 K-Medoid

The K-medoid algorithm is similar to the K-means already described. The main difference is that, rather than using the centroid as the mean for each cluster, the method uses the element with smallest distance to the centroid as the mean. This new approach decreases the method sensibility to outliers [Han et al., 2011].

The algorithm starts by selecting $k$ medoids from the set of all elements present on the data, one for each cluster. This step, similarly to K-means, can be done randomly or using more sophisticated approaches. Each element is then associated to the cluster which it has the smallest distance to the medoid. The quality of each configuration is measured by a cost function, similar to the Loss function of K-means.

$$C = \sum_i ||X_i - \mu_{z_i}||^2 \tag{2.16}$$

where $X$ represents the set of points and $\mu_{z_i}$ the medoid of the cluster where point $i$ was placed.

Whenever a new element is added to the cluster, the medoid is changed to the new element, and the new cost of the cluster is measured. The swap of the medoid is undone if the total cost of the configuration increases.

A simple algorithm for the K-medoids can be seen below:

**Algorithm 1** K-Medoids Algorithm

```
Require: $\mathcal{D}$                                                  ▷ Data Points
Require: $k$                                           ▷ number of medoids (clusters)
 1: Select $k$ points in $\mathcal{D}$ as the medoids M.
 2: for Each Data point $d_i$ $\mathcal{D}$ do
 3:     Associate each $d_i$ to the closest in M.
 4: end for
 5: while The configuration cost $C = \sum_i ||\mathcal{D}_i - \mu_{z_i}||^2$ decreases do
 6:     for Each medoid $m$ in $M$ and each non-medoid $d_i$ in the same cluster as M do
 7:         Swap $m$ and $d_i$
 8:         Computate the new configuration cost $C$.
 9:         If the total cost increases, undo the swap
10:     end for
11: end while
```

In this work we decided to use the K-Medoids because it supported, in addition to the traditional way of passing the data points as the input data, the use of only a matrix of distances of these points. Instead of giving the position in space for each element, we only have the distances between one and the others in our method.

For our problem, a element is a feature of the data, and each row of the matrix is a vector representing the distances, between this feature and all the others. We will enter in more details on how this was done in Chapter 5 of this work.

# Capítulo 3

# Related Works

In this chapter we present an exploratory review of past work focusing directly on view generation and multiview learning. We cover manual view generation approaches [Dalip et al., 2015], some automatic view generation methods [Kumar & Minz, 2016; Sun et al., 2011] and other topics whose ideas were helpful to our work, like Feature Subset Ensembles (FSE) [Ho, 1998] and feature selection [Oliveira et al., 2003].

Learning with multiple distinct feature sets was first proposed by [Blum & Mitchell, 1998], introducing the concept of *co-training.* The motivation was to take advantage of the consensus of different and independent views of a same problem, and also to improve performance by co-training two classifiers. The focus of their work was to allow inexpensive unlabeled data to augment a much smaller set of labeled examples. Basically, each classifier was trained on a different view and the most confident predictions on unlabelled data were used by them to train each other. An experimentation was conducted on classifying Computer Science webpages, comparing the co-training method against a supervised approach that uses only the labeled data. The results showed the potential of multiview and cotraining, with co-training having better results on all three different classifiers used. After that, many other works have addressed this problem with an array of theoretical developments and successful applications.

It is important to mention a different approach that can be related to the multiview method, that is the use of multiple ensembles of features to help on the learning task, which was called FSE (Feature Subspace Ensemble). One of the first proposes was the Random Subspace Method (RSM) proposed by [Ho, 1998], which creates each feature subset using random features. The method was created to construct 'Decision Forests', which are basically a set of different trees used together in a classification task. The authors showed with experiments the efficiency of their approach compared to single decision trees and other forest constructions methods, and also showed an interesting aspect of this way of generating trees, they had a smaller agreement ratio compared to other methods. This aspect is interesting because it allow less redundant information and, in a certain way, is similar to a multiview approach (different views tend to have different kinds of information).

The idea of creating random subspaces was explored in other scenarios, not only in 'decision forest' generation, and evolved into creating subsets with highly discriminative and distinct features using, for example, feature selection techniques as used in [Oliveira et al., 2003]. The FSEs approaches can be related to the soft aspect of our proposed soft Views, given that a set of features allow features that were already present in another set to be used. Even then, it has a significant difference. In this work we are trying to explore, first and foremost, the concept of a view. It is the exact opposite of the subsets from the FSE methods. We try to explore the benefits of grouping similar features to find specific and significant information from each group, while the other approaches only explicit explore groups with dissimilar and low correlated features. Basically, FSEs methods, in general, disregard the potential benefits of searching for views.

With a different focus than the issue of automatic view generation, the authors in [Dang & Croft, 2010] proposed a feature selection method where, at each step, the algorithm randomly selects a group of features and then tries to improve it by removing or adding a new feature. At the end of each iteration, the method creates a set of features that can not overlap with other sets. While it was not the intention of the authors, this step can easily be seen as a multiview generation process where, at each iteration, a view of features is created. The rest of the process goes more in the direction of feature selection, by using only some features of each group in a final classification. Even then, the process of indirectly creating views is worth mentioning. It is also relevant to show that the approach presented comparable or even better performances, on a ranking task, than learning from the original features. In the remaining of this chapter we will focus on the multiview learning methods most similar to our proposed approach.

A set of works that clearly showed the potential of multiview learning, and inspired this work, were [Dalip et al., 2012, 2013, 2017], in which a multiview application to the problem of document quality assessment was proposed, using datasets based on information gathered from social websites like Wikis (e.g Wikipedia) and Question&Answer boards (e.g StackOverFlow). The method presented by the authors achieved state-of-the-art performance by adopting a multiview approach based on complex engineered features, organized into natural views and combined with the original individual features. Each natural view was based on an aspect of the data, be it the length of the text, the style and the structure used by the author when writing, or even the edit history of the community article. After each selected feature was associated to one view, all the views were used to evaluate the documents presented, and they separated scores were then combined to get a final and more precise result. This work is a clear example of the use of the multiview learning to solve a real problem. This work presented exciting results, that showed better performance compared to state-of-art single view classifiers in the datasets evaluated. The authors did not, however, address the possibility of using an automatic view generation approach to automatically generate the views given their set of features, which was one of the motivations of this dissertation.

Other applications of multiview learning were presented in recent works, such as [Chikhi, 2016], which focuses on multiview clustering, [Zhao et al., 2018] that successfully applies a manifold learning approach on a multiview scenario, and [Cano, 2017] that deals with a multi-instance learning problem using a multiview approach. These studies have successfully applied the multiview method presenting results with significant gains over the state-of-art approaches. However, they too have not addressed the possibility of automatically generating the views used in their multiview methods.

[Sun et al., 2011] proposed an approach to the problem of creating artificial views by applying genetic algorithms (GA) to multiview semi-supervised learning. They used GA to find two promising feature subsets that would be retained as views, if they reached maximum classification agreement. In experiments with single-task and multi-task support vector machines, they were able to confirm the effectiveness of the proposed view construction method.

The authors in [M. Chen & Chen, 2011] extended co-training to cases where no explicit views are available. In particular, they proposed an algorithm that splits the feature space during learning into two mutually exclusive subsets, explicitly encouraging the co-training to be successful. Their method outperformed previous results in an object recognition task. In a following work [Gao & Yang, 2014], they adapted their approach to the learning to rank task. During the ranking process, their method automatically splits the feature set into distinct views. Then, it minimizes the combined likelihood rank loss of the views on the labelled instances (a common list-wise ranking technique) while maximizing the rank agreement between them on the unlabelled instances. This approach was presented on semi-supervised and cross domain learning, and it outperformed a baseline using natural splits. However, the method proposed by the authors is dependent on the ranking process, which restricts it only to to the learning to rank domain, so we can not opt to only generate the artificial views. Our approach, while being tested only on the quality assessment scenario, can be used in any other problems with no limitations.

The authors in [Di & Crawford, 2012] studied the automatic generation of views not on text data, but for the task of hyperspectral image classification. They investigated three important aspects of multiview learning: (1) The aggregation of features according to similarities to promote view diversity using clustering; (2) The use of feature space bagging and random selection to explore the feature space and avoid uninformative or corrupted views; and (3) the uniform split of data across the space to ensure view sufficiency. They observed that increasing the number of views and its randomness leads to more diversity and less noise, with an improved performance. This aspect is somehow explored in our IKMVG approach, as the use of our soft views allow an almost unlimited number of views. (But always trying to filter and avoid non informative views). [Di & Crawford, 2012] work was not used as a baseline for our method given that it was built for an active learning scenario, which is different from our goal. Their algorithm is based on the changes provided by the new learned examples to select the best approach to partition the feature set, so if we can not have this changes on our approach (which is the case, as we have a fixed training and test set during all the process), it can not be applied in our scenario. Even so, we still acknowledge the value of the ideas presented and use some similar strategies in our method.

Finally, on a recent work, the authors of [Kumar & Minz, 2016] did an overview of the state-of-art in automatic view generation and proposed a method called Optimal Feature Set Partition (OFSP), which is the closer to our approach yet. Their method has no limits on the number of views (many other approaches, such as [M. Chen & Chen, 2011] or [Sun et al., 2011] only use 2 views) and they iteratively try to construct and improve already existing views. OFSP, at each step, selects a candidate feature on a pool of unused features and tries adding it to all already built views. The views are compared to each other, and the new best view is evaluated.

The method proposed by [Kumar & Minz, 2016] presented improved results when compared to the others methods proposed in the literature [Bryll et al., 2003; Ho, 1998; Sun et al., 2011; Di & Crawford, 2012]. Because of that, we are going to consider [Kumar & Minz, 2016] our baseline. Their method is presented on Algorithm 2

Unlike the previous methods mentioned, in our approach, we adopt soft views instead of mutually exclusive subsets. In our experiments, the view consistency rule was proving to be a detriment to the learner and the final result, and by using our so called soft approach we managed to achieve considerably better results compared to the strict approach. In addition, our method proposes that we retain the original features in the final set of features and views, while filtering potentially irrelevant

**Algorithm 2** OFSP Algorithm

```
Require: $\mathcal{D}_{train}$                ▷ training set with $n$ features and $y$ labels.
Require: $\mathcal{D}_{val}$                  ▷ training set with $n$ features and $y$ labels.
Require: $k$                                  ▷ number of views
 1:
 2: for $i = 1$ to $k$ do
 3:     $\mathcal{V}_i \leftarrow \varnothing$          ▷ Creating k-number of NULL views
 4:     $\epsilon^i \leftarrow 0$          ▷ Initializing classification accuracy of the classifier
 5: end for
 6:
 7: for $feat = 1$ to $n$ do
 8:     for $i = 1$ to $k$ do
 9:         $\mathcal{V}_{temp} \leftarrow \mathcal{V}_i \cup v_{feat}$          ▷ Adding feature $v_{feat}$ to the $\mathcal{V}_i$ view
10:         $\epsilon^i_{temp} \leftarrow$ EVALUATE($\mathcal{X}_{temp}, \mathcal{D}_{train}, \mathcal{D}_{val}$)
11:         $\epsilon^i_{diff} \leftarrow \epsilon_{temp} - \epsilon^i$
12:     end for
13:     if $max(\epsilon^i_{diff}) > 0$ then
14:         $t = argmax(\epsilon^i_{diff})$          ▷ $t^{th}$ view which has $\epsilon^i_{diff} > 0$
15:         $\mathcal{V}_t \leftarrow \mathcal{V}_t \cup v_{feat}$
16:         $\epsilon^t \leftarrow \epsilon^t_{temp}$
17:     end if
18: end for
19: return $\mathcal{V}_{opt} = V_1, X_2, ..., V_k$
20:
21: function EVALUATE($\mathcal{F}$, $\mathcal{D}_{train}$, $\mathcal{D}_{pred}$)
22:     Train the learner on $\mathcal{D}_{train}$ using features $\mathcal{F}$
23:     return Accuracy evaluation on $\mathcal{D}_{pred}$
24: end function
```

views using a feature selection approach. This way of combining features was proved interesting in some experiments of [Dalip et al., 2015] and during our internal tests as well. Finally, our method IKMVG has another differential in how we explore the feature space to automatically generate the views. We try to explore both similar and dissimilar relationships between features, to create stronger and diverse views. In the next chapter we will go a bit in depth on the manual views used by [Dalip et al., 2015] and on this work and then explain our method and experimental results that justify the benefits of our approach.

# Capítulo 4

# Manual Views

In this work we compare our approach to that of [Dalip et al., 2012, 2017], where views were generated manually. This was done because one of our main objectives was to see if an automatic method could generate equally good or similar views compared to manually created views. In [Dalip et al., 2015] the author described in details how was the process of selecting and grouping the features into views, with strong experimental results to back it up. This work gave us an interesting starting point for our work now.

The authors [Dalip et al., 2012, 2017] collected a high number of quality metrics from Wikis websites and used them to create features to describe the instances of each Wiki document. In order to generate views manually, the quality features were grouped according to their sources. For example, features that used text readability as their main source of information were all grouped in the same view. Using this intuitive scheme, the authors of [Dalip et al., 2012, 2017] grouped 68 features into 6 different types of views — (1) Structure, (2) Readability, (3) Length, (4) Style, (5) Edit History, and (6) Article Graph. The results of their work highly surpassed the performance of other approaches and are, from what we know, the state-of-art in the area of Automatic Quality Assessment. In the following, we present these views proposed by them in more detail.

## Structure

Structure features indicate how well the article is organized. According to Wiki quality standards, a good article must be organized such that it is clear, visually adequate, and provides the necessary references and pointers to additional material. Thus, features derived from the article structure are used, in an attempt to describe its section organization, and its use and distribution of images, links, and citations. Details of these features can be seen in Table 4.1.

| **Structure** | |
|---|---|
| *ts-abslen* | Length of abstract |
| *ts-avsecl* | Average section length |
| *ts-avparl* | Average paragraph length |
| *ts-avsubps* | Average subsections per sections |
| *ts-cite* | n. of citations (references) |
| *ts-citplen* | n. of citations divided by text length |
| *ts-citpsec* | ratio between n. of citations and sections |
| *ts-imgps* | n. of images per section |
| *ts-lnkpl* | n. of links per text length |
| *ts-minsecl* | Length of shortest section |
| *ts-maxsecl* | Length of largest section |
| *ts-secs* | Section Count |
| *ts-subsec* | Sub-section Count |
| *ts-stdsecl* | Section length standard deviation |
| *ts-xlnks* | n. of links to external sources |
| *ts-xlnkps* | n. of external links per section |

**Tabela 4.1.** Structure Features

## Readability

Readability features are intended to estimate the age or US grade level necessary to comprehend a text. The intuition behind these features is that good articles should be well written, understandable, and free of unnecessary complexity. To accomplish this, these features use several well-known readability indexes. Details can be seen in Table 4.2.

| **Readability** | |
|---|---|
| *tr-ari* | Automated Readability Index [Senter & Smith, 1967] |
| *tr-flesh* | Flesch reading ease [Flesch, 1948] |
| *tr-fog* | Gunning Fog Index [Gunning, 1952] |
| *tr-kincaid* | Flesch-Kincaid [Ressler, 1993] |
| *tr-liau* | Coleman-Liau [Coleman & Liau, 1975] |
| *tr-lix* | Läsbarhets index [Björnsson, 1968] |
| *tr-smog* | Smog-Grading [Mc Laughlin, 1969] |

**Tabela 4.2.** Readability Features

## Length

Length features are indicators of the article size. The general intuition behind them is that a mature and good quality text is probably neither too short, which could indicate an incomplete topic coverage, nor excessively long, which could indicate verbose content. Further, in Wikis, stub articles (draft quality) are expected to be short, which reinforces the correlation between length and quality. Details of these features can be seen in Table 4.3.

| **Length** | |
|---|---|
| *tl-charcnt* | Character count |
| *tl-phrcnt* | Phrase Count |
| *tl-wordcnt* | Word count |

**Tabela 4.3.** Length Features

## Style

Style features are intended to capture the way the authors write the articles through their word usage. The intuition behind them is that good articles should present some distinguishable characteristics related to word usage, such as short sentences. Details of these features can be seen in Table 4.4.

| **Style** | |
|---|---|
| *ty-auxverb* | n. of auxiliary verbs |
| *ty-conj* | n. of words that are conjunctions |
| *ty-lgphra* | Size of the largest phrase |
| *ty-nomina* | n. of nominalizations |
| *ty-passive* | n. of passive voice sentences |
| *ty-plgphr* | %p where (length - avg. length) >= 10 words |
| *ty-prepo* | n. of prepositions |
| *ty-prono* | n. of pronouns |
| *ty-psmphr* | %p where (avg. length - length) >= 5 words |
| *ty-questn* | n. of questions |
| *ty-sartic* | #p starting with an article |
| *ty-sconj* | #p starting with a conjunction |
| *ty-sintp* | #p starting with an interrogative pronoun |
| *ty-sprepo* | #p starting with a preposition |
| *ty-sprono* | #p starting with a pronoun |
| *ty-ssubcnj* | #p starting w/ a subordinating conjunction |
| *ty-tobe* | n. of uses of verb “to be” |

**Tabela 4.4.** Style Features

## Edit History

Edit History indicators have been mainly used to estimate the maturity level of the content. In general, a content that received many edits has likely improved over time. Details features can be seen in Table 4.5.

| **Edit History** | |
|---|---|
| *r-3month* | proportion of r-rcount in last 3 months |
| *r-activeu* | reviews by top-5% most active reviewers |
| *r-age* | Article age |
| *r-ageprev* | ratio between article age and n. of reviews |
| *r-anonym* | n. of reviews made by anonymous users |
| *r-discuss* | n. of posts on the article's discussion page |
| *r-modline* | % of lines of curr. version $\neq$ from reference |
| *r-occasion* | reviews by reviewers with less than 4 edits |
| *r-probrev* | ProbReview |
| *r-rcount* | Review count |
| *r-reguser* | n. of reviews made by registered users |
| *r-revpday* | percentage of reviews per day |
| *r-rperusr* | ratio between r-rcount and n. of reviewers |
| *r-stdrevu* | std. dev of r-rperusr |

**Tabela 4.5.** Edit History Features

## Article Graph

Finally, the Graph indicators are those extracted from the links between articles (citations). The main motivation for using them is that citations between articles can provide evidence about their importance. Details can be seen in Table 4.6.

| **Article Graph** | |
|---|---|
| *n-assortii* | Assortativity In-In |
| *n-assortio* | Assortativity In-Out |
| *n-assortoi* | Assortativity Out-In |
| *n-assortoo* | Assortativity Out-Out |
| *n-cluster* | Clustering coefficient |
| *n-idegree* | n. of citations of an article from other ones |
| *n-linkcnt* | n. of links to articles (even if not written) |
| *n-odegree* | n. of links to other articles |
| *n-pgrank* | Pagerank value of an article |
| *n-reciproc* | Reciprocity |
| *n-translat* | n. of article versions in other languages |

**Tabela 4.6.** Article Graph Features

# Capítulo 5

# Method

In this chapter, we present in details our complete multiview approach. We first describe our multiview framework used to evaluate the performance of any set of views in a given dataset, and, afterwards, our proposed method to automatically generate views, which is called Iterative K-Medoids View Generator (IKMVG).

## 5.1 Multiview Evaluation Framework

Is this work, we adopt the same multiview framework proposed in [Dalip et al., 2012, 2013, 2017] and illustrated in Figure 5.1. We made this decision given that that work serves as a start point for our approach and so we would be able to do fair comparisons between the results given by both methods. That framework assumes that a natural split is already available for the feature set, such that it is possible to map a feature to a view.

As shown in Figure 5.1, the multiview evaluation framework can be divided into two levels. At *level 0*, each view $v$ is evaluated, using all instances $i$ in the dataset, but using only those features that belong in view $v$. Thus, for each view $v$, a classifier $V_v$ will provide a score $e_{vi}$ for each instance $i$. In order to create the learner for the view $v$ we have access to a training set in which each instance $i$ is represented by the features of the view $v$.

At *level 1*, the scores $e_{vi}$ are used as a new feature for the level 1 classifier, to predict the final score. As seen in Figure 5.1, these scores can be used in isolation (MVIEW) or combined with the original features (MVIEW+F0). To accomplish this, we create a level-1 learner by performing cross validation in the training set obtaining the estimates $e_{vi}$ of each instance in the training set. As in Dalip et al. [2012, 2013, 2017], we use the same machine learning model in both levels — Support Vector Re-

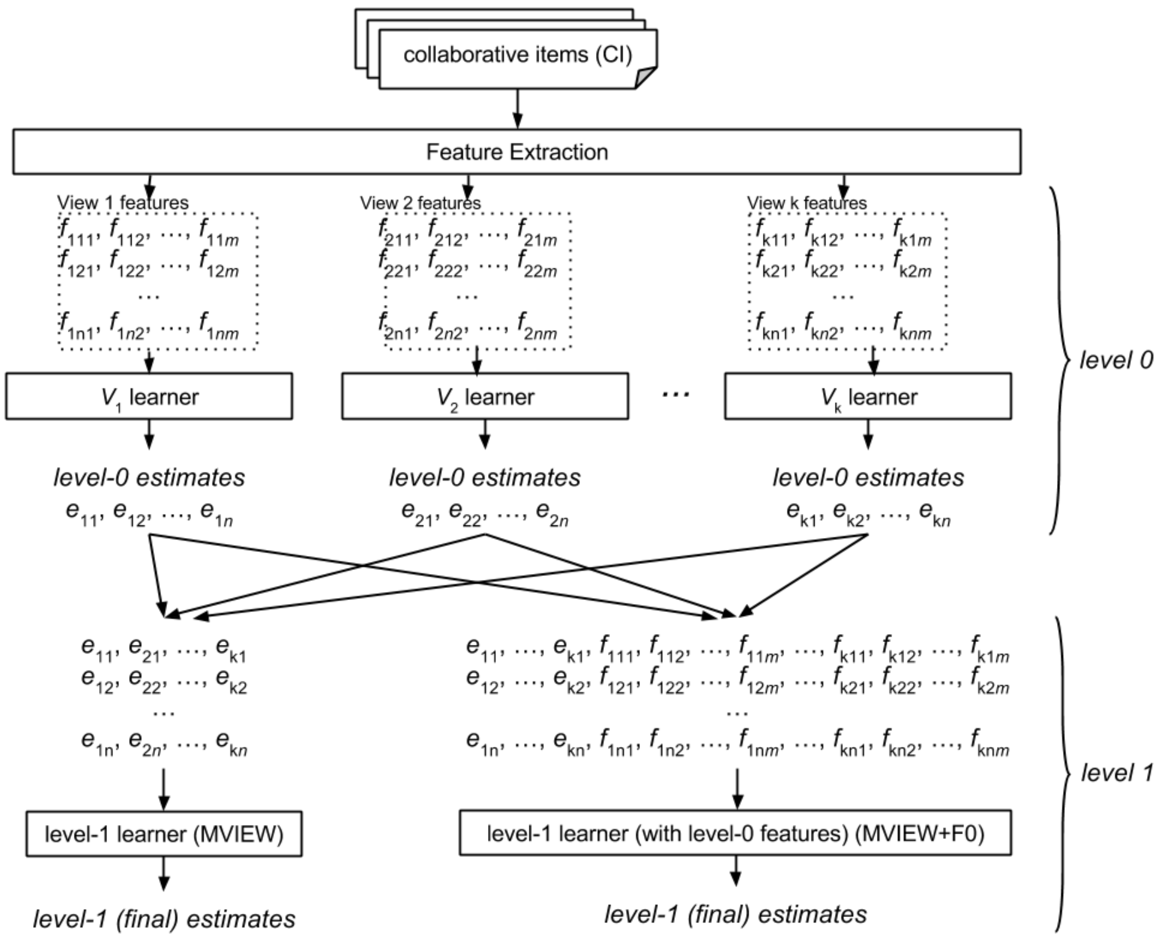

**Figura 5.1.** Illustration of two alternative multiview quality assessment methods (MVIEW and MVIEW+F0). Each $V_v$ learner corresponds to a view $v$, and $f_{vij}$ is the feature $j$ of instance $i$, according to view $v$. Finally, $e_{vi}$ represents the prediction on level-0 learning of each view $v$ for the instance $i$. The value $e_{vi}$ is later used as a feature on the level-1 learning.

gression. This way we can have a fair comparison of our new results by generating artificial views against the natural views select by an expert in the area.

## 5.2 View Generator

In this section we present our proposed method, called Iterative K-Medoids View Generator (IKMVG), and describe the intuition and history behind its development. A terminology that will be used in this thesis are the concepts of 'good' and 'bad' views. in our work, a 'good' view is the one that brings relevant information to our model, improving its overall result. On the other hand, a 'bad' view only brings noisy and redundant information to the model. This means it will both increase its complexity and decrease its effectiveness.

### 5.2.1 Intuition and Evolution

The main intuition behind our method is that, if two features are highly correlated to each other, then they are probably similar or refer to the same subject and should be grouped in the same view. Our first step into this direction was to calculate the Spearman Correlation Coefficient of all pair of features, and with this information we applied a clustering algorithm to group them into views. While the result using only this approach was not satisfactory for us. Some views were interesting and close to those in the manual set, but others were bad and the overall result of the combination of the views was inferior to the manual.

One idea to improve the views and consequently the final result was to change the number of clusters (and thus views) created by our clustering algorithm. We first started using the same number as the manual approach (6), and then tried smaller and bigger values. Unfortunately, neither variation had a significant and positive impact. Using a small number of clusters resulted in losing to much of the overall information of the data and a higher number of artificial views meant that the features were too spread and it was hard to get information about the relationships of the features.

To face the problem of the features becoming too spread with a high number of views, we proposed a strategy called ‘soft views’. It is a loosening on the view consistency rule that claimed that each feature could be part of one exclusive view. This rule did not allow us to create a high number of views without losing the relationships between features. Our ‘soft views’ essentially meant that one feature could be part of an indefinite number of views, with no limitation. This may look like a conflict to the definition of views and multi-view itself, as it originally tries to create the most independent views as possible. Our reasoning is that the features on different views, even if separated by an expert, may still not be completely independent and have some valuable relationships to each other. Consequently, the created views were not fully independent from the start, and creating ‘hard’ partitions would result in a loss of valuable information.

Another possible discussion is that if our method, given the new ‘soft views’, can still be classified as a multi-view approach. It is a hard and interesting question, and figure 5.2 illustrates our point of view about it. While our method is not exactly a multi-view approach, given that we relax the view consistency rule, it is not a bagging approach either, as it is not a random process, but a more guided one (based on similarity functions). Our approach seems to be in a middle ground between a multi-view and a bagging scenario. The closer we are from hard partitioning and a semantic division, the closer we get from a multi-view method. On the other hand, the closer we get to a random process, the closer we are to a bagging method.

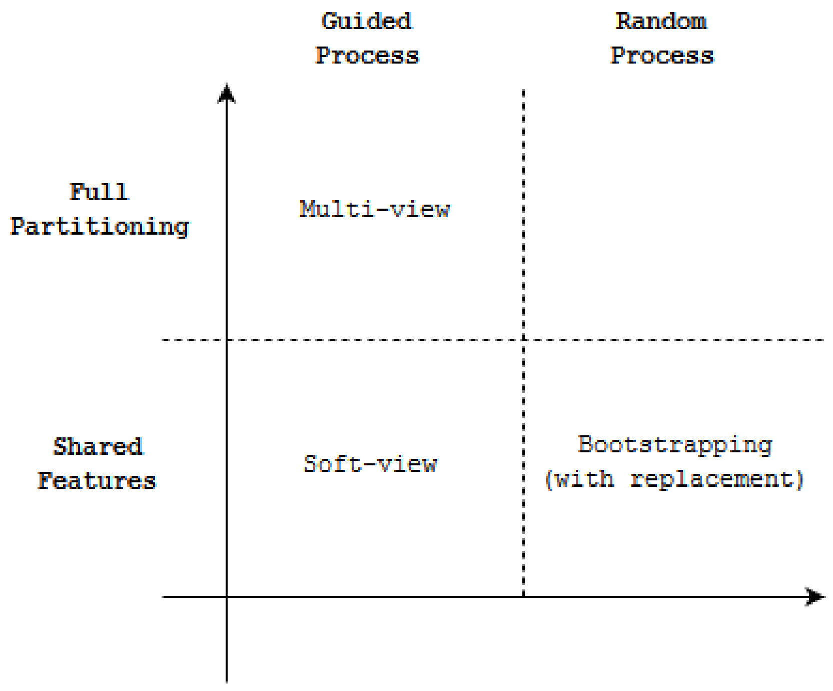

**Figura 5.2.** Illustration classifying our new soft-views, comparing it with a bagging and a multi-view approach in terms of how the feature set partitioning is done.

Given our new ‘soft-views’, our new problem was on how to use it and to use it wisely. We propose to explore the soft views in an iterative approach. We select, at each iteration, a random subset of features, based on the original set of features. Then, we use a clustering algorithm to new create views based on similarity metrics between the features selected. In the end we combine all the views created during the whole process.

By selecting a random subset of features at each time we avoid the problem of the clustering algorithm only generating the same clusters. To use this idea wisely and avoid generating bad views or even a really high number of views, we evaluate our set of views created during each time, and only kept the ones that improved our result. By combining all of this ideas we ended on a algorithm called IKMVG, that we present next.

### 5.2.2 IKMVG

To generate views we propose the *Iterative K-Medoids View Generator* method (IKMVG), described in Algorithm 3. IKMVG is an iterative method where, in each iteration, a random set of features is used to produce a new view through a clustering procedure. The views produced are then added to the final set of views only if they contribute to improve the overall system performance.

The IKMVG algorithm starts by setting a threshold $\epsilon_{best}$ for the quality of the results by evaluating the system using only the set of individual features (line 2). Afterwards, at each iteration, IKMVG creates $k$ view candidates, using a random sample of the original features (lines 4–5). All candidates ($\mathcal{V}_{candidates}$) are then combined with the already accepted views ($\mathcal{V}$) and the original features ($\mathcal{F}_0$), to be evaluated as input for the multiview learner, through the function *Evaluate* (line 6).

The candidates are retained in the final pool of views if they are able to reduce the learning error. Otherwise, they are discarded so that the method avoids retaining bad views (lines 7–10). Note that, in a same iteration, the candidate views are mutually exclusive. However, since any feature can be considered in a future iteration, the views in the final set can overlap.

An example of the execution of one iteration of the algorithm on a dataset with 5 original features can be seen in Fig 5.3

In function GetViews, new views are generated by clustering similar groups of features, through the *k-medoids* clustering algorithm[MacQueen et al., 1967]. To estimate the similarity between features, we used the absolute value of the Spearman Correlation Coefficient (SCC). As shown in Geng et al. [2007], it is more effective to calculate the correlation between features using the classification errors they produce, instead of the actual feature values. The idea here is to classify the data using only a single feature, and then use its classification result as the feature value instead. Thus, to obtain the SCC, we perform a cross-validation test on the training dataset, using each feature as an independent regressor. Each feature is then represented by the set of errors yielded in each fold and the SCC is computed.

The computed SCCs are stored in a correlation matrix $M$, where $M_{ij}$ is the unsigned correlation between features $i$ and $j$[1]. Matrix $M$ is then used by $k$-medoids to split the features into clusters. Each cluster will correspond to a view. In addition, we also use matrix $M' = 1 - M$, a *dissimilarity* matrix, to capture interactions among dissimilar features. As described in Algorithm 3 (lines 20–26), each time we call the

[1]We ignore positive or negative correlations, considering only the degree of correlation between features.

**Algorithm 3** IKMVG Algorithm

```
Require: $\mathcal{D}_{train}$                                  $\triangleright$ training set
Require: $\mathcal{D}_{val}$, $\mathcal{D}_{test}$              $\triangleright$ validation and test sets
Require: $\mathcal{F}_0$                                        $\triangleright$ original feature set
Require: $N$                                                    $\triangleright$ number of Iterations
Require: $k$                                                    $\triangleright$ number of views per Iteration
Require: $\rho$                                                 $\triangleright$ percentage of sampled features per iteration
Require: $fs$                                                   $\triangleright$ percentage of features to keep on feature selection

1: $\mathcal{V} \leftarrow \varnothing$                          $\triangleright$ Set of views
2: $\epsilon_{best} \leftarrow$ EVALUATE($\mathcal{F}_0$, $\mathcal{D}_{train}$, $\mathcal{D}_{val}$)
3: for $N$ iterations do
4:     $\mathcal{F}_{sample} \leftarrow$ GETSAMPLE($\mathcal{F}_0$, $\rho$)
5:     $\mathcal{V}_{candidates} \leftarrow$ GETVIEWS($\mathcal{F}_{sample}, k$)
6:     $\epsilon_{new} \leftarrow$ EVALUATE($\mathcal{F}_0 \cup \mathcal{V} \cup \mathcal{V}_{candidates}$, $\mathcal{D}_{train}$, $\mathcal{D}_{val}$)
7:     if $\epsilon_{new} < \epsilon_{best}$ then
8:         $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{V}_{candidates}$
9:         $\epsilon_{best} \leftarrow \epsilon_{new}$
10:    end if
11: end for
12: $F_{selected} \leftarrow$ FEATURESELECTION($\mathcal{F}_0 \cup \mathcal{V}$, $fs$)
13: $\epsilon_{final} \leftarrow$ EVALUATE($F_{selected}$, $\mathcal{D}_{train}$, $\mathcal{D}_{test}$)

14: function EVALUATE($\mathcal{F}$, $\mathcal{D}_{train}$, $\mathcal{D}_{pred}$)
15:     Train the learner on $\mathcal{D}_{train}$ using features $\mathcal{F}$
16:     return Error evaluation on $\mathcal{D}_{pred}$
17: end function

18: function GETVIEWS($\mathcal{F}, k$)
19:     $M \leftarrow$ SPEARMANCORRELATIONMATRIX($\mathcal{F}$)
20:     if ComputeSimilarity=1 then
21:         $ComputeSimilarity \leftarrow 0$
22:         return KMEDOIDSCLUSTERING($M, k$)
23:     else
24:         $ComputeSimilarity \leftarrow 1$
25:         return KMEDOIDSCLUSTERING(1-$M, k$)
26:     end if
27: end function
```

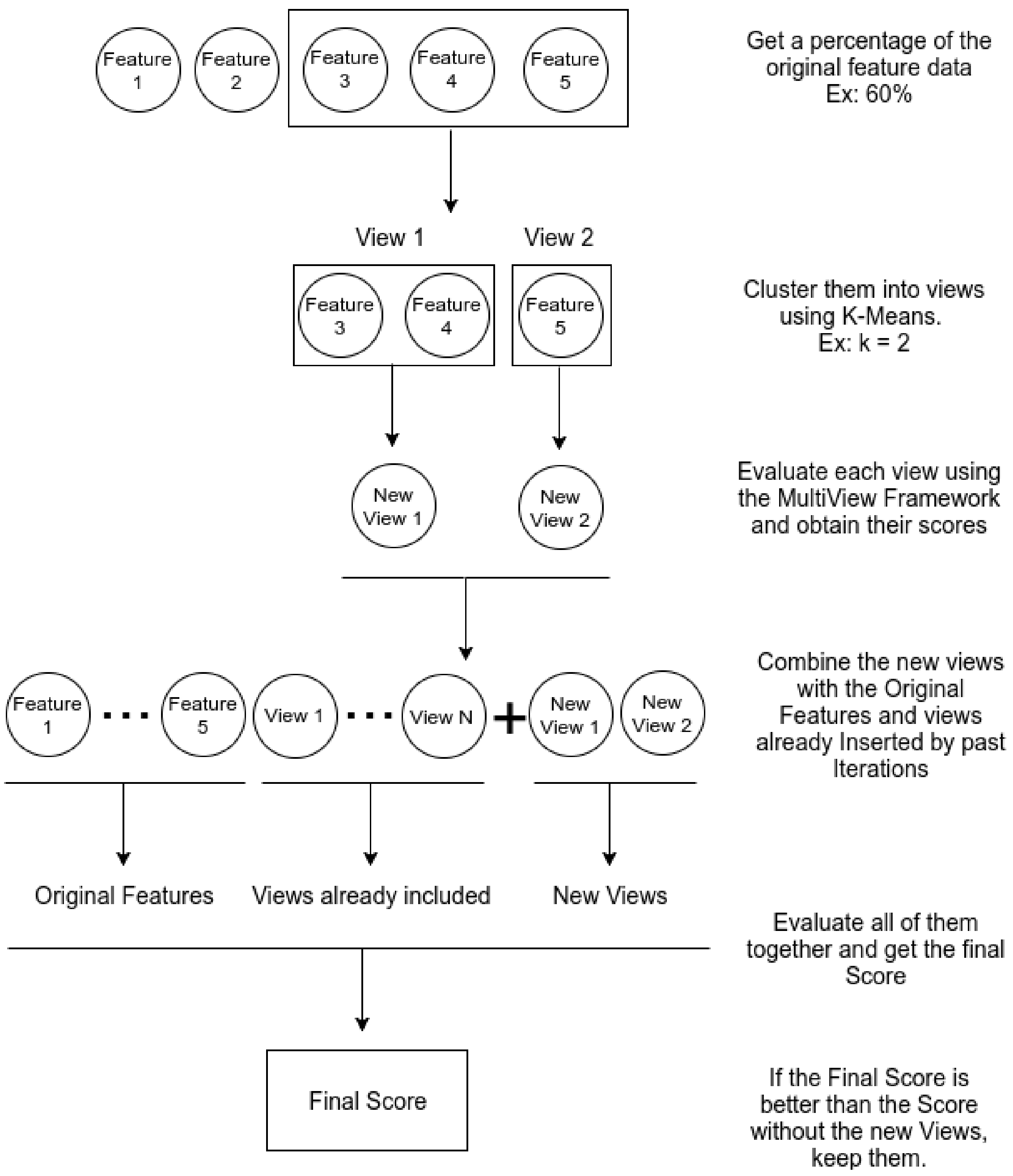
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Get a percentage of the original feature data Ex: 60%
View 1
View 2
Feature 3
Feature 4
Feature 5
Cluster them into views using K-Means. Ex: k = 2
New View 1
New View 2
Evaluate each view using the MultiView Framework and obtain their scores
Feature 1
Feature 5
View 1
View N
New View 1
New View 2
Combine the new views with the Original Features and views already Inserted by past Iterations
Original Features
Views already included
New Views
Evaluate all of them together and get the final Score
Final Score
If the Final Score is better than the Score without the new Views, keep them.

**Figura 5.3.** Illustration of one iteration of our IKMVG algorithm, using a data with 5 original features and with already some views included.

function $GetViews$, we generate views using the similarity matrix (if the previous call used the dissimilarity matrix) or the dissimilarity matrix (if the previous iteration used the similarity matrix). This alternating strategy aims at not favoring $M$ or $M'$ during the automatic generation of views.

Finally, function FeatureSelection (lines 28–30) is used to filter and remove potentially noisy and irrelevant views or features that may be harmful to the learner. This is done by applying the InfoGain Feature Selection [Shannon, 2001] method on our final set of views ($\mathcal{F}_0 \cup \mathcal{V}$) so that only $fs\%$ of the total set of features will remain for the final evaluation. The way we apply the feature selection algorithm on our final set is by considering each view as a new feature of our data (as seen in the Level 1 learner on Figure 5.1). This process will generate a rank based on relevance for each view/feature and then maintain only the most relevant ones. This is an important step given that our method, depending on the number of iterations, can generate a large number of views.

It is important to mention that the original features ($\mathcal{F}_0$) are used in all iterations of the IKMVG, but, during the evaluation, they can be removed to analyze the power of only the automatically generated views. We use the original features in IKMVG to guarantee a good starting point for our method. Assuming we had a fresh start, with no features whatsoever at the beginning, we would not be able to evaluate the generated views and judge if we should add them or not. This would allow the algorithm to add noisy views right from the start. Another favourable point of using original views is to allow our method to reach a higher score faster, using less iterations.

Another important discussion is the reasoning behind using matrices $M$ and $M'$, and not only $M$, as expected on a multiview approach. The results of only grouping similar features (according to SCC) were not satisfactory, as they were worse compared to the manual approach. The overall result was improving slowly during the iterations, given that many of the views created started to being very similar to each other. The same phenomenon happened while using $M'$, and thus we decided to apply both matrices in the algorithm to increase variability and improve the growth speed of the method's result.

# Capítulo 6

# Experiments

In this chapter, we present our test datasets, empirical methodology and results of our experiments.

## 6.1 Experimental Setup

### 6.1.1 Datasets

In this work we considered three datasets — Wikipedia, StarWars and Muppet — also used in [Dalip et al., 2012, 2017], for the task of quality assessment. The Wikipedia dataset corresponds to a sample of pages (including links and review history pages) extracted from the Wikipedia online encyclopedia[1], English edition, in 2010. It is the world largest online encyclopedia. The StarWars dataset is a sample of the Wookieepedia encyclopedia[2], the largest hosted in the Wikia service. It is a thematic encyclopedia focused on the Star Wars universe. The Muppet dataset is also an encyclopedia provided by the Wikia service[3], focused on the American TV show "The Muppet Show". In Table 6.1 it is possible to find additional information about the datasets used, as such as the number of features, number of documents in total, number of total classes and the machine learning task approached by each of them.

### 6.1.2 Evaluation, Algorithms and Procedures

As in [Dalip et al., 2012], for all quality assessment datasets (Wikipedia, Wookieepedia, and Muppet), we evaluate their effectiveness using the Mean Squared Error measure

[1]`https://en.wikipedia.org/`
[2]`http://starwars.wikia.com/`
[3]`http://muppet.wikia.com/`

| Dataset | Number of Documents | Learning Task | Number of class labels | Number of features | Number of views |
|---|---|---|---|---|---|
| Wikipedia | 2,964 | Quality regression | 6 Classes (int) | 68 | 6 |
| StarWars | 1,302 | Quality regression | 3 Classes (int) | 68 | 6 |
| Muppet | 1,395 | Quality regression | 1 to 5 Stars (float) | 68 | 6 |

**Tabela 6.1.** Test datasets used in our experiments.

(MSE). MSE is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{q}_i - q_i)^2 \tag{6.1}$$

where $\hat{q}_i - q_i$ corresponds to the estimate error calculated as the absolute difference between the quality value predicted ($\hat{q}_i$) and the true quality value $q_i$. The number of predictions is represented by $n$. Given the nature of the method, we are then looking for the lowest MSE as possible, which would mean that the predicted values are the closest possible (in average) to the correct values.

To ensure the generalization of our results, we use 5-fold cross-validation, with validation sets [Kuhn & Johnson, 2013]. In this procedure, the dataset is split into five partitions such that, at each run, one partition is used as test set, one as validation set, and the remaining as training set. The final performance measurement is taken as the average of the results observed for all the runs. We also assess the statistical significance of our results by means of a paired t-test with 95% confidence and Holm correction [Holm, 1979] to account for multiple tests. This test assures that the best results, marked in **bold**, are statistically significant. The obtained results (and their 95% confidence intervals) are described in Section 6.2.

The multiview framework described in Section 5.1 uses two learners: level 0 and level 1. Our experiments use the LIBSVM implementation [Chang & Lin, 2011] of the Support Vector Regression (SVR), with a radial-basis function kernel, in both cases. The SVRs were parametrized using the training set, with the values of $C$, $\epsilon$ and $\gamma$ under the intervals $C = [2^{-5}, 2^{-3}, ...., 2^{13}, 2^{15}]$, $\epsilon = [0.1]$, $\gamma = [2^{3}, 2^{1}, ...., 2^{-5}, 2^{-7}]$.

Regarding our algorithm, IKMVG, we used 20 iterations on all datasets. Other parameters, such as the percentage of features to be sampled per iteration ($\rho\%$), the percentage of features to keep on feature selection ($fs$), and the number of clusters for k-medoids ($k$), were obtained by using grid search cross-validation on the training set.

As a byproduct of our algorithm we ran an experiment using the six most similar views generated by the IKMVG compared to each one of the manual views, as an attempt to use exactly the same number of manual views. Our tests presented interesting results, where the manual views had no statistically significant results, based on our evaluated metrics, to the six most similar automatic generated views.

All experiments were run on a Quad-Core Intel® Xeon® E5620, running at 2.4GHz, with 32Gb RAM. In order to consider the cost of all processing power of each approach, we report the sum of "user time" and "system time" consumed by the process execution in all experiments regarding execution time.

### 6.1.3 Influence of number of clusters and percentage of features

It is important to mention the impact of the variations of $\rho\%$ and $k$ and which values where used as inputs to the cross-validation.

The higher the value of $\rho\%$ is, the higher are the numbers of features used and, if the value is 100%, we would get the original separation obtained by using k-medoids in all the data (because then we are not removing any features from our input). This way, really high values are not interesting because they impact the diversity of the views, as that the closer we get from 100% the closer we get to the same division on each iteration. On the other hand, small values are usually not interesting either, the reason being that the clusters would have a really small amount of information, with a small quantity of features on each cluster, which makes it difficult to obtain a strong views.

Analyzing the number of clusters $k$, the higher its value is, the higher is the probability of having clusters with a smaller size, as the number of features is fixed (on each execution), and only highly similar features would be put together.

High values of $k$ did not perform well in our experiments too. It resulted, in the end of all iterations, in a really high number of views, in which many of them were not good (The views were redundant and did not bring any new information). Another problem was that the higher the number of clusters, the higher is the cost of the algorithm, given that the learner must be executed for each cluster separately on the fist step of the Multiview FrameWork.

Given this experimentation and analysis, we decided for 3 different values for $\rho\%$ and for $k$. The end values are $\rho\% = [30, 50, 70]$ and $k = 3, 4, 5$. This avoids really small or really high values for $\rho\%$ and high values of $k$, because of the problems already mentioned. The final combination used on each final fold of the 5-fold cross-validation was, as mentioned, decided by using another cross-validation on the training set varying the combinations of $\rho\%$ and $k$.

## 6.2 Experimental Results

In this section, we present our results, in particular, the effectiveness of automatically generated views, their discriminative power and similarity when compared to manual views.

### 6.2.1 Effectiveness of Automatically Generated Views

On this first experiment we evaluate the effectiveness of our Automatic Generated Views and compare them to the original set of features and the manual views for our three datasets. We start by noticing that our experimental results in Table 6.2 provide evidence towards the benefits of using the multiview framework (considering either manually created views or the views produced by IKMVG) to reduce the error produced by the predictions using only the original set of features $\mathcal{F}_0$ in most cases. In this scenario, $\mathcal{F}_0$ is a non-multiview method which uses all the level 0 original features as a feature set and then apply the SVR method, as proposed in [Dalip et al., 2011]. For this experiment we removed the original features from the final IKMVG set, and only evaluated the test set with our automatically generated views.

The use of manual views reduces the error on WIKIPEDIA and STARWARS, but manual views obtain not statistically significant results with the original features on the MUPPET dataset. Given these results, it is expected that the automatic method could yield better results compared to the single-view approach too. In fact, the use of automatically generated views is always superior to the set $\mathcal{F}_0$, reducing the error in 7%, 20% and 2% on WIKIPEDIA, STARWARS and MUPPET, respectively. It is interesting to note that our method, IKMVG, surpasses even the manual views in some datasets, which shows that even manually created views may not be 100% accurate and have room for improvement.

| METHOD | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| $\mathcal{F}_0$ | $0.898 \pm 0.019$ | $0.082 \pm 0.004$ | $1.751 \pm 0.045$ |
| IKMVG | $0.838 \pm 0.019*$ | $0.060 \pm 0.003*$ | $1.686 \pm 0.053*$ |
| MANUAL | $0.864 \pm 0.016$ | $0.058 \pm 0.003*$ | $1.753 \pm 0.050$ |

**Tabela 6.2.** Average MSE results of manual views and automatically generated views using the IKMVG method. Asterisks mean statistically significant differences (p-value $< 0.05$) over the values without asterisk on the same dataset.

These results show that despite carefully grouped by an expert, the manual views may prevent the learning method from capturing important relationships between features from different views, which reduces the capability of discovering discriminative patterns. Basically, by restricting a feature to only one view, this strict view partitioning approach is limiting the search space of the learner algorithm and losing efficiency because of it. Conversely, IKMVG is a data-driven method, which does not rely on experts, but tries to automatically generate views that can reduce the regression error. Our method does not rely on a stricted partitioning of the data, so our search space is bigger, allowing the learner to find more useful patterns and relationships between features thus improving our results. The higher number of possibilities that IKMVG gives is probably one of the main reasons that we were able to reach better results when compared to the manual approach. In all evaluated experiments, as mentioned before, the results obtained with automatically generated IKMVG views are never statistically worse than the results of manual views, which motivates the adoption of automatically generated views that rely on finding close groups of features automatically discovered from the data.

| METHOD | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| $\mathcal{F}_0$ | $0.898 \pm 0.019$ | $0.082 \pm 0.004$ | $1.751 \pm 0.045$ |
| IKMVG+$\mathcal{F}_0$ | $0.821 \pm 0.018*$ | $0.064 \pm 0.004*$ | $1.706 \pm 0.047*$ |
| MAN+$\mathcal{F}_0$ | $0.831 \pm 0.018*$ | $0.065 \pm 0.004*$ | $1.743 \pm 0.044$ |

**Tabela 6.3.** Average MSE results of the original features $\mathcal{F}_0$ and the combination of $\mathcal{F}_0$ with manual views and automatically generated views using the IKMVG method. Asterisks denotes statistically significant differences.

We also evaluate the combination of original features with multiviews, since they present the best results in previous works in some datasets [Dalip et al., 2012, 2017]. Table 6.3 shows that the combinations of views with original features (represented in the table as IKMVG+$\mathcal{F}_0$ and MAN+$\mathcal{F}_0$) are capable of slightly improving some of the previously described multiview results. The reason for this improvement may lie on a loss of information when using only views. When using a multiview approach, in the end, each view becomes a new feature and the learner is trained with these features. Given that the manual approach and ours both generate less 'end features' than the original method, some information is probably missed. This way, combining the views with the original features may solve this problem and allows the learner to obtain the information. Table 6.3 also shows that the combination IKMVG+$\mathcal{F}_0$ presents better results than $\mathcal{F}_0$, as well as statistically superior results compared with the combination MAN+$\mathcal{F}_0$ on MUPPET. These results provide further evidence towards the benefits of using the proposed automatic views, since besides not relying on specialized effort to define views, they consistently improve the results of $\mathcal{F}_0$.

To justify the better performance of IKMVG, as mentioned before, our method

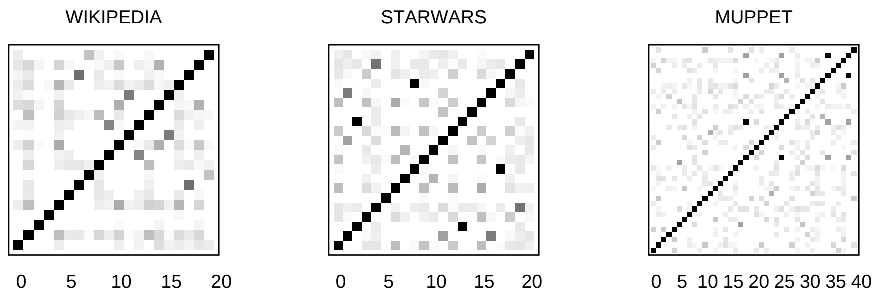

**Figura 6.1.** Feature overlapping in automatically generated views in WIKIPEDIA, STARWARS and MUPPET datasets. Each cell $(i,\ j)$ in the colormap corresponds to the proportion of features shared by views $v_i$ and $v_j$. A white color indicates no overlapping whereas a black color indicates maximum overlapping.

allows the learner to explore a larger space of information, so we have potentially more useful information to help the learner take better decisions. Moreover our views are already being tested during their generation, so we can probably assure a good result if the generation step was successful.

An important point to mention is the reason why the performance of MAN+$\mathcal{F}_0$ and IKMVG+$\mathcal{F}_0$ is worse than their counterparts that did not used $\mathcal{F}_0$ in STARWARS. In [Dalip et al., 2012, 2017] it is reported that the 'length' manual view basically solves this dataset quality assurance problem on its own. A high number of documents on this wikia are called stubs, with represent unfinished texts about a certain topic. Given their nature, they determined quality is low, and this aspect can easily be detected by the size of the text. It means that the majority of the features are only generating noise for the learner, and so adding the original set is harming our final result.

### 6.2.2 Feature Sharing on Automatically Generated Views

In this next experiment, presented in Figure 6.1, we had the objective to study the behavior of our automatically generated soft views, and if it is worth increasing the cost of the learning process by adding more views. We want to see if there is redundant and shared information between the views. In our experiments, during one execution of the IKMVG algorithm, we generated 20, 21 and 40 soft views for WIKIPEDIA, STARWARS and MUPPET, respectively. As observed in Figure 6.1, all the generated views shared features with, at least, one other view. In average, they shared 6% of the features in WIKIPEDIA, 7% in STARWARS, and 4% in MUPPET. Given these percentages, our approach is not creating too much redundant views, as the average of shared features between views is small. Given this fact, we can confirm that we are widing the range of our learner, and giving it the opportunity to discover unseen relationships between features. As shown in Table 6.2, these views reduced the MSE error when compared with the manual views in 3%, and 2% for WIKIPEDIA and MUPPETS datasets, respectively. This makes us believe that, in fact, adding more views, even if they are allowed to share some features, can help the learner to obtain more useful information and reach a better result.

### 6.2.3 Discriminative Power of Automatically Generated Views

To evaluate the discriminative power of the views generated by IKMVG, we evaluate the Information Gain (IG), based on Shannon Entropy [Shannon, 2001], yielded by the automatically generated views, the manually described views, and the original features of our datasets. In our context, the information gain is a measure of the reduction in entropy of the relevance response variable, after the value for the feature (or view) is observed, which provides evidence for the discriminative power of the evaluated feature. In this experiment, for each dataset, we added together the original set of features and both manual and automatic generated views. We rank the views and original features according to their Information Gain and analyse the proportion of views generated automatically by our IKMVG method that are in the top most informative. Table 6.4 presents the results considering the top 1 to 10 more informative features or views.

| Top Rank | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| 1 | 1.0 | 1.0 | 1.0 |
| 2 | 1.0 | 1.0 | 0.5 |
| 3 | 1.0 | 1.0 | 0.3 |
| 4 | 0.8 | 1.0 | 0.3 |
| 5 | 0.8 | 1.0 | 0.4 |
| 6 | 0.8 | 1.0 | 0.3 |
| 7 | 0.9 | 0.9 | 0.4 |
| 8 | 0.8 | 0.8 | 0.4 |
| 9 | 0.8 | 0.8 | 0.3 |
| 10 | 0.7 | 0.7 | 0.4 |

**Tabela 6.4.** Proportion of automatically generated views in the top most informative.

In all datasets, the most discriminative information is provided by an automatically generated view. Moreover, the top 5 most informative features or views in WIKIPEDIA and STARWARS are all in the proposed automatically generated views. Even considering all the top 10 most informative views and features, the views generated by IKMVG are also among the most frequent on the top-ranked information. Note that in MUPPETS the performance of our proposed views is not so good. One of the hypothesis for it is the fact that, as mentioned in [Dalip et al., 2015], MUPPETS ratio system is not trustworthy, given that it is only star-based and the number of users votes on some pages is really small. even then, our method was still able to improve the results of the original set of features and the manual views. These results demonstrate the discriminative power of our artificial views, which surpasses the discriminative power of the manually generated views and of the original features.

The main reasons behind these results are probably the freedom during the view creation and that our views were already evaluated during the IKMVG algorithm. In more details, first the freedom is meant to be that our approach allow more feature combinations in a view, compared to the manual ones. By using two different metrics (simillarity or dissimilarity matrix), we are exploring a bigger space and thus opening new possibilities to find strong views. Secondly, after being generated, our automatic views are already tested, and only used if they improved the training set. This step ensures that the final view set will have, for the most part, good and high informational views.

A deeper study on the percentage of automatic informational views X automatic non informational views, based on their position on the rank would be an important future analysis, that may allow us to reduce a bit the number of generated views without impacting the efficiency of the method, thus gaining time performance.

### 6.2.4 Similarity between Manual Views and Automatically Generated Views

Table 6.5 presents the similarity between a manually generated view and its most similar artificial view. The main point here is that, given the fact that the manual views yield good results and that our method still explores the similarity between features, we have a good chance to generate some views similar to the manual generated ones.

In order to perform this experiment, we see each view as a *set of features* and then compare views using the Jaccard similarity coefficient. In most cases, the similarity is below 0.7, which indicates that the automatically generated views are not quite similar to the manual ones. This can be explained by the fact that our method drives a sample of all the features (30%, 50% or 70%) before splitting this sample into views. Therefore, instead of generating one single view related to a specific manually defined view, in most cases IKMVG generates multiple views that correspond to a set of correlated features. This would be a bad approach if feature sharing between views were not allowed, because some complementary relationships of similar features would not be found if they were in different views. IKMVG, however, allows it to happen and to even improve the overall result, by using its soft views approach.

Among all the datasets, the automatically generated views for WIKIPEDIA were among the closest to the manual ones. In fact, the WIKIPEDIA is the one with higher number of documents, and so this dataset provides more training examples, which improves the quality of our automatically generated views. Particularly, the Readability view was actually reconstructed by our automatic method, since the metrics for Readability are very similar, due to their exploitation of similar evidences (such as number of words or characters), and cluster naturally. Some of the views generated by the proposed IKMVG are also close to the manual views Edit History and Article Graph.

Following the WIKIPEDIA trend, on STARWARS our method was capable of finding views close to Edit History, Article Graph and Readability. We lack a similar view to the Length one, and this is expected given that this is the smallest view (only 3 features) and usually our artificial views are bigger.

In MUPPET we did not generate any highly similar views (The closest was on Edit History). This result is again expected, since the natural views do not perform well on this dataset and, thus, we expected the automatically generated views to contain significant differences.

| Manual View | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| Structure | 0.41 | 0.27 | 0.39 |
| Readability | 1.00 | 0.71 | 0.29 |
| Length | 0.33 | 0.33 | 0.33 |
| Style | 0.44 | 0.29 | 0.29 |
| Edit History | 0.63 | 0.42 | 0.44 |
| Article Graph | 0.69 | 0.50 | 0.36 |

**Tabela 6.5.** Jaccard similarity between each manual view and its most similar automatically generated view.

To further illustrate the view similarities, Tables 6.6 and 6.7 shows, for each of the six manual views on the WIKIPEDIA dataset, the most similar artificial one. Each column corresponds to one artificial view, and the marks indicate that a feature is included in that view. This results provide evidence that automatically generated views can be highly similar, or even equal (view 2), to the manual views. Particularly, view 1 is clearly related to the Edit History view, but also contains features from the Article Graph view, which demonstrates a proximity between these two manually defined views. Artificial view 2 corresponds exactly to the Readability view. View 3 presents various features from Structure, Length, and Style features, showing that

| | Feature/View | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Edit History | r-3month | ✓ | | | | ✓ | |
| | r-activeu | ✓ | | | | | |
| | r-age | | | | | | |
| | r-ageprev | ✓ | | | | | |
| | r-anonym | ✓ | | | | | |
| | r-discuss | ✓ | | | | | |
| | r-modline | ✓ | | | | | |
| | r-occasion | ✓ | | | | | |
| | r-probrev | ✓ | | | | | |
| | r-rcount | ✓ | | | | | |
| | r-reguser | | | | | | |
| | r-revpday | | | | | | |
| | r-rperusr | ✓ | | | | | |
| | r-stdrevu | | | | | | |
| Article Graph | n-assortii | | | | | ✓ | |
| | n-assortio | | | | | ✓ | |
| | n-assortoi | | | | | ✓ | |
| | n-assortoo | | | | | ✓ | |
| | n-cluster | | | | | | |
| | n-idegree | ✓ | | | | ✓ | |
| | n-linkcnt | | | | | ✓ | |
| | n-odegree | | | | | ✓ | |
| | n-pgrank | | | | | | |
| | n-reciproc | | | | | ✓ | |
| | n-translat | ✓ | | | | ✓ | |

**Tabela 6.6.** Representation of the six automatically generated views that are the most similar to manual views on WIKIPEDIA. Each column correspond to an automatic view and a ✓ indicates the presence of the feature on that view. Edit History and Article Graph views are presented

these three manually defined views are very close to each other. Finally, views 4, 5 and 6 are clearly associated with Length, article Graph and Structure, respectively.

Despite the small differences between each manual view and its most similar artificial view, this small subset of artificial views can provide the same discriminative information as the manual views. Table 6.8 presents the MSE results obtained with manual views and its six corresponding artificial views. The MSE results are statistically tied on all datasets, providing evidence that our method is capable of obtaining a subset of views that are very close and as effective as the manually defined ones. This result is interesting because one can argue that there is a problem comparing the IKMVG with the manual views, given that they are always at an small number and thus can give less information. In the presented experiment the same number of views

|  | Feature/View | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Readability | tr-ari |  | ✓ |  |  |  |  |
|  | tr-flesh |  | ✓ |  |  |  |  |
|  | tr-fog |  | ✓ |  |  |  |  |
|  | tr-kincaid |  | ✓ |  |  |  |  |
|  | tr-liau |  | ✓ |  |  |  |  |
|  | tr-lix |  | ✓ |  |  |  |  |
|  | tr-smog |  | ✓ |  |  |  |  |
| Style | ty-auxverb |  |  | ✓ |  |  |  |
|  | ty-conj |  |  |  |  |  |  |
|  | ty-lgphra |  |  | ✓ |  |  |  |
|  | ty-nomina |  |  | ✓ |  |  |  |
|  | ty-passive |  |  | ✓ |  |  |  |
|  | ty-plgphr |  |  | ✓ |  |  |  |
|  | ty-prepo |  |  | ✓ |  |  |  |
|  | ty-prono |  |  |  |  |  |  |
|  | ty-psmphr |  |  | ✓ |  |  |  |
|  | ty-questn |  |  |  |  |  |  |
|  | ty-sartic |  |  | ✓ |  |  |  |
|  | ty-sconj |  |  | ✓ |  |  |  |
|  | ty-sintp |  |  | ✓ |  |  |  |
|  | ty-sprepo |  |  |  |  |  |  |
|  | ty-sprono |  |  |  |  |  |  |
|  | ty-ssubcnj |  |  | ✓ |  |  |  |
| Len | tl-charcnt |  |  |  | ✓ |  |  |
|  | tl-phrcnt |  |  | ✓ |  |  |  |
|  | tl-wordcnt |  |  | ✓ |  |  |  |
| Structure | ts-abslen |  |  |  |  |  | ✓ |
|  | ts-avsecl |  |  | ✓ |  |  | ✓ |
|  | ts-avparl |  |  | ✓ |  |  |  |
|  | ts-avsubps |  |  |  |  |  |  |
|  | ts-cite |  |  | ✓ |  |  | ✓ |
|  | ts-citplen |  |  |  |  |  |  |
|  | ts-citpsec |  |  | ✓ |  |  |  |
|  | ts-imgps |  |  |  |  |  | ✓ |
|  | ts-lnkpl |  |  |  |  |  |  |
|  | ts-minsecl |  |  |  |  |  |  |
|  | ts-maxsecl |  |  | ✓ |  |  |  |
|  | ts-secs |  |  |  |  |  |  |
|  | ts-subsec |  |  |  |  |  |  |
|  | ts-stdsecl |  |  | ✓ |  |  |  |
|  | ts-xlnks |  |  | ✓ |  | ✓ | ✓ |
|  | ts-xlnkps |  |  |  |  |  | ✓ |

**Tabela 6.7.** Representation of the six automatically generated views that are the most similar to manual views on WIKIPEDIA. Each column correspond to an automatic view and a ✓ indicates the presence of the feature on that view. Readability, Style, Length and Structure views are presented

were used, which avoids this possible criticism.

| METHOD | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| Six Views | 0.857±0.015 | 0.061±0.005 | 1.770±0.081 |
| MANUAL | 0.864±0.016 | 0.058±0.003 | 1.753±0.050 |

**Tabela 6.8.** Average MSE results of the manual views and six artificial views of IKMVG which are the most similar to the manual views.

In addition to evaluating the nearest artificial view to each manually defined view, we have also examined the distribution of all artificially generated views to evaluate if our views present a global tendency to fit into the manually defined ones. In order to represent the various automatically generated views, we have clustered them (using $k$-Means) into 6 groups. We then evaluate how close the views on each cluster are from the 6 manually defined views.

| | Clusters of IKMVG Views | | | | | |
|---|---|---|---|---|---|---|
| Edit History | **83**% | 29% | 4% | 13% | 7% | 9% |
| Structure | 5% | **42**% | 10% | 4% | 21% | 18% |
| Readability | 2% | 0% | **49**% | 0% | 8% | **22**% |
| Style | 0% | 11% | 6% | **36**% | **25**% | 19% |
| Length | 1% | 7% | 12% | 26% | **25**% | 14% |
| Article Graph | 8% | 12% | 19% | 21% | 14% | 18% |

**Tabela 6.9.** Proportion of features from manually labelled views on six clusters of IKMVG views. Each column corresponds to a cluster of similar automatically generated views on the WIKIPEDIA dataset.

Table 6.9 presents the proportion of features[4] on each cluster of automatically generated views, on the WIKIPEDIA dataset. The most common kind of view in each cluster is shown in bold face. The first column of the table corresponds to the only cluster which is closely related to the specific manually defined view Edit History. This fact provides evidence for the idea that Edit History is an important view, with highly correlated features and, at the same time, very different from features in other views.

Other automatically generated views tend to contain features from two or more of the manually labelled views. Some of the clusters of automatically generated views are closer to manual views, like Readability with 49% and Structure with 42%, which provides evidence for their importance. However, most features from the manual views are more spread over the clusters of artificial views, since features from different views are closely related by the Spearman correlation, or provide informative relationships

[4]The proportion is weighted according to the number of features from each manually labelled view.

based on their supplementary discriminative information. Another important aspect to mention is the use of the dissimilar matrix for the k-medoids, which helps to create more diverse views with not so similar features together.

It is also interesting to analyse the correlation between the most discriminative views, according to IG, on each dataset. To this effect, we evaluate the proportion of features from each manually defined view present on the most discriminative artificial views. Table 6.10 shows that, for all datasets, the most discriminative views are not biased to any specific manual view. In fact, the most discriminative views are the combination of features from different manual views, indicating that there is important discriminative information to be obtained from the relationship of dissimilar features associated with different manual views. This further amplifies the importance of using both similar and dissimilar matrix during our algorithm, to create a more diverse set of views.

| | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| Structure | 29% | 24% | 20% |
| Readability | 13% | 19% | 20% |
| Length | 8% | 3% | 1% |
| Style | 21% | 19% | 20% |
| Edit History | 24% | 17% | 7% |
| Article Graph | 5% | 17% | 32% |

**Tabela 6.10.** Proportion of features from manually labelled views on the most discriminative artificial views generated for each dataset.

## 6.3 Efficiency of IKMVG compared to FSE methods

In this section we present a comparison between our proposed IKMVG method against one of the FSE methods, in particular the Random Subspace Method [Ho, 1998]. The idea behind this experiment is to evaluate our approach against another automatic approach that explores a concept similar to our soft views. In RSM it is, like our approach, possible to have one feature in more than one view. The RSM method uses 2 parameters, $n$ and $f$, where $n$ is the number of feature subsets (views) that will be generated and $f$ is the number of features on each subset.

The RSM method, for this experiment, was set with $n = 20$ to generate 20 feature subsets and $f = 20$ to generate 20 random features on each of them. This parameters were choosen to approximate to the number of views generated by IKMVG and the

**Tabela 6.11.** RSM results compared to IKMVG and the MANUAL views. Asterisks denotes statistically significant differences.

| Metodo | WIKI | SW | MUP |
|---|---|---|---|
| IKMVG | $0.836 \pm 0.019*$ | $0.060 \pm 0.003$ | $1.686 \pm 0.053*$ |
| RSM | $0.869 \pm 0.021$ | $0.060 \pm 0.004$ | $1.697 \pm 0.064$ |

average number of features on each view. This way we can have a fair judgment in comparing both methods without one method having access to alot more or alot less information. The results are presented in Table6.11.

The results in the STARWARS dataset were similar in both methods, but our approach beats the RSM in both MUPPETS and WIKI with statistically significant results. Specially in WIKI, this is a really positive result for us, given that this dataset is the most complete one and with the more number of data. Another conclusion that can be extracted from here is the importance of using strong and meaningful views. RSM generate views with random features, while IKMVG uses the concept of similarity to group features. This, together with that fact that our views are already being evaluated during the algorithm execution, are probably the main reasons of our better results. The results presented showed that our method can compete with manual views or even artificially created ones, and to have a good performance against both methods. As a future work we intend to test our method against a more complex automatic view generator.

## 6.4 Effectiveness of Automatically Generated Views compared to other automatic method

In this section we compare our IKMVG approach to one of the literature state-of-art approaches to automatic multi-view generation. The idea here is to show that our method have an equal or even better effectiveness compared to other methods presented before. We will compare the IKMVG against the OFSP presented in [Kumar & Minz, 2016], focusing on effectiveness and time performance.

### 6.4.1 Comparison on Effectiveness

Table 6.12 presents the results comparing both approaches on our three datasets.

The results presented in Table 6.12 show that our approach is better or equal to OFSP in all datasets evaluated. In particular, IKMVG results in WIKIPEDIA were better than the baseline. This is particularly intersting since WIKIPEDIA has

| METHOD | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| IKMVG | $0.838 \pm 0.019*$ | $0.060 \pm 0.003$ | $1.686 \pm 0.053$ |
| OFSP | $0.861 \pm 0.016$ | $0.060 \pm 0.003$ | $1.713 \pm 0.050$ |

**Tabela 6.12.** Average MSE results of the automatically generated views using the IKMVG method and using the OFSP method. Asterisks denotes statistically significant differences.

a more complete manual rating system which, because of that, is more difficult to predict classes than our STARWARS dataset, as we can see in the MSE result. On STARWARS and MUPPET our method is equally effective comparing to the baseline.

| METHOD | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| IKMVG+$\mathcal{F}_0$ | $0.821 \pm 0.018*$ | $0.064 \pm 0.004*$ | $1.706 \pm 0.047*$ |
| OFSP+$\mathcal{F}_0$ | $0.849 \pm 0.018$ | $0.073 \pm 0.004$ | $1.737 \pm 0.044$ |

**Tabela 6.13.** Average MSE results of the combination of $\mathcal{F}_0$ features with IKMVG generated views and with OFSP generated views. Asterisks denotes statistically significant differences.

By combining the features from IKMVG and FSOP with the original dataset we arrived at the results presented in table 6.13. Our method is significantly better than OFSP in all three datasets in this tests. While this combination improved the results of both methods on WIKIPEDIA, it led to worse results on STARWARS and MUPPETS. This can be explained by the fact that the original features add noise to the data on these two datasets, which had an even bigger negative impact on OFSP.

### 6.4.2 Comparison on Time consumption

Regarding the execution time, OFSP tries, at each step, to insert a random feature in all the views that already exist, keeping only the one that most improves the results. This approach makes the algorithm have a high computational cost, given that each feature will be evaluated once on each existing view. More specifically, each evaluation corresponds to the execution of a regression method (in our case, the SVR) which is the most computationally expensive procedure in the method. Given $v$ views generated by OFSP and $n$ features on the dataset, the OFSP method performs $n * v$ evaluations. Differently, the number of evaluations performed by IKMVG does not depend on the number of features, which makes the cost much smaller since our method requires less evaluations than the OFSP. In fact, our approach performs $N * v$ evaluations using the SVR, where $v$ is the number of views and $N$ is the number of iterations. We also execute the k-medoids method on each iteration ($N$ times). However the execution time of k-medoids is not significant when compared to the time spent to perform the SVR evaluations.

Table 6.14 presents a comparision of the time spent to run one fold on each dataset. The use of $N = 20$ iterations to execute IKMVG can make this method achieves more effective results than OFSP, which relies on the $n = 68$ features to perform extra evaluations that not only do not contribute to the OFSP effectiveness, but also make the OFSP method less efficient than our proposal. Moreover, the higher the number of features added on each view generated by OFSP, the more expensive are the newer evaluations, given that the SVR learner will need to process more information from the additional features inside each view. Unlike IKMVG, the OFSP efficiency and scalability can be substantially affected by a high number of features.

| METHOD | WIKIPEDIA | STARWARS | MUPPET |
|---|---|---|---|
| IKMVG | 26m02s | 7m30s | 9m22s |
| OFSP | 713m40s | 289m21s | 312m40s |

**Tabela 6.14.** Comparison on the computational time consumption between IKMVG and OFSP methods

The results in table 6.14 were expected, given our previous analysis, but it is important to mention that the difference between the two approaches was even bigger given that IKMVG was optimized, avoiding to redo already calculated views in many cases, while OFSP was implemented exactly as it was explained in [Kumar & Minz, 2016], with no optimizations whatsoever.

# Capítulo 7

# Conclusions

In this work we have proposed an automatic view generator for the task of quality assessment in user generated content. In particular, we adopted an iterative approach that automatically generates and evaluates *soft views* for the data, i.e. views containing overlapping sets of features. Apart from not relying on experts to group features into views, our method can be used in any scenario, without domain restrictions and without knowing the semantic meaning of each feature.

Based on the work by [Dalip et al., 2015], we evaluate our method in three different Wiki datasets, generating artificial soft views for each of them and comparing the results obtained by these views to the state-of-art. Our experimental results provide good evidence towards the benefits of using our automatically generated views. In fact, these are capable of reducing the prediction error of the original features by around 20% on one of the datasets, giving better and statistically significant results in the three of them.

Our method also configures a competitive alternative to the manual views, since it can automatically generate views very close to those manually defined, and even reduce the prediction error of the manual views, in some scenarios. Compared to the 6 manual ones, we could replicate exactly one of them for the Wiki dataset and many others were really close by the jaccard distance. Moreover, using the six most similar views created during our experiments we were able to get a result comparable to the manual one.

## 7.1 Contributions Callback

The main contribution of this work is the proposal of an innovative method for generating views for multiview learning, that is automatic, efficient, and potentially domain independent. Being automatic means that it does not need an expert to make the decision of which features will belong to each views, which can be a hard and time consuming task. Our experiments proved the good performance of our method by showing that it can compete really well against the manually created views, that, in the scenario evaluated, could be considered the optimal views, as they were carefully created by and expert. Finally, given that our method does not depend on any semantic meaning of the features, it can potentially be applied in any scenario where a multiview approach seems interesting. In addition to our main contribution, we helped the development of the multiview research area in many aspects. The first one was the idea of using 'soft views'. While the idea of creating distinct groups of subsets of features with feature repetition is nothing new, the idea was never explored in a multiview scenario, where we seek to group similar and correlated features together.

Second, by realizing a big analysis and experimentation on our automatically generated views and comparing them to the manual ones we presented the weak and strong points of our method in the domain of quality assessment on the datasets evaluated.

Finally, even though we initially thought to group only similar features, after observing how FSE methods work, and how distinct and dissimilar features been put together can improve the overall result, we decide to join these two ideas. Our method tries to combine the multiview hypothesis with the assumption that its good to group similar features [Dalip et al., 2015] with ideas proposed in [Oliveira et al., 2003], where the focus is in grouping dissimilar features. Our approach was able to achieve strong results that were able to beat both the manual approach by [Dalip et al., 2015] and the automatic approach of [Kumar & Minz, 2016] in at least one of the datasets evaluated. We also had significantly better results compared to the basic FSE approach called RSM presented in [Ho, 1998]

## 7.2 Future Work

As future work, we intend to evaluate our proposal in other domains. Our approach has the potential to show good results independent of the domain, because it only needs information from the values of the features , which are available in any machine learning domain. We do not need to know the meaning of the feature or if it is a dataset that has a clear feature separation into views. Given this, we will also apply the method to datasets where features are harder to group into meaningful views, such as image collections where the images are represented using pixels.

In parallel to explore multiple domains, we also intend to improve the efficiency of our method and study other approaches to group the views, besides the one we proposed based on the Spearman correlation. There are a high number of metrics that can be used in the literature and maybe they can have better results compared to the Spearman in specific scenarios. We also would like to measure the impact of these metrics in the result. In recent experiments we empirically found that, by using random values on the matrices $M$ and $M'$, we achieved a close final result compared to what we reported. Given that, understanding how much a metric is helping would help a lot to improve the performance in that specific scenario.

Another experiment we plan to do in the future is to use our method with learning algorithms that have a higher performance than SVR. The baseline result from a Random Forest, for example, surpasses the results of the SVR in all the datasets presented on this study. In this work we did not use RFs because of how complex it would be to understand the real value of our method, given that a RF already has an "internal multiview", where each tree only receives part of the features. Now, that we assure the value of our approach, it would be interesting to see how much it could enhance the already strong result of the Random Forest algorithm. Another interesting idea is to add a bias on the tree generation step of the Random Forest algorithm. This way, the we could generate trees based on our similarity and dissimilarity metrics, exploring the ideas proposed for our view generation IKMVG.

# Referências Bibliográficas

Björnsson, C.-H. (1968). *Lesbarkeit durch Lix.* Pedagogiskt centrum, Stockholms skolförvaltn.

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. Em *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pp. 92--100, New York, NY, USA. ACM.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5--32.

Bryll, R.; Gutierrez-Osuna, R. & Quek, F. (2003). Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291 – 1302. ISSN 0031-3203.

Cano, A. (2017). An ensemble approach to multi-view multi-instance learning. *Knowledge-Based Systems*, 136:46 – 57. ISSN 0950-7051.

Chang, C.-C. & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chikhi, N. F. (2016). Multi-view clustering via spectral partitioning and local refinement. *Information Processing and Management*, 52(4):618--627. ISSN 0306-4573.

Coleman, M. & Liau, T. L. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

Dalip, D. H.; Goncalves, M. A. & Cristo, M. (2015). *A MULTI-VIEW APPROACH FOR ASSESSING THE QUALITY OF COLLABORATIVELY CREATED CONTENT ON THE WEB 2.0.* Tese de doutorado, Graduate Program in Computer Science of the Universidade Federal de Minas Gerais.

Dalip, D. H.; Gonçalves, M. A.; Cristo, M. & Calado, P. (2011). Automatic assessment of document quality in web collaborative digital libraries. *J. Data and Information Quality*, 2(3):14:1--14:30. ISSN 1936-1955.

Dalip, D. H.; Gonçalves, M. A.; Cristo, M. & Calado, P. (2012). On multiview-based meta-learning for automatic quality assessment of wiki articles. Em *Proceedings of the Second International Conference on Theory and Practice of Digital Libraries*, TPDL'12, pp. 234--246, Berlin, Heidelberg. Springer-Verlag.

Dalip, D. H.; Gonçalves, M. A.; Cristo, M. & Calado, P. (2013). Exploiting user feedback to learn to rank answers in q&#38;a forums: A case study with stack overflow. Em *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pp. 543--552, New York, NY, USA. ACM.

Dalip, D. H.; Gonçalves, M. A.; Cristo, M. & Calado, P. (2017). A general multiview framework for assessing the quality of collaboratively created content on web 2.0. *Journal of the Association for Information Science and Technology*, 68:C1–C1, 271–539.

Dalip, D. H.; Lima, H.; Gonçalves, M. A.; Cristo, M. & Calado, P. (2014). Quality assessment of collaborative content with minimal information. Em *IEEE/ACM Joint Conference on Digital Libraries, JCDL 2014, London, United Kingdom, September 8-12, 2014*, pp. 201--210. IEEE Computer Society.

Dang, V. & Croft, B. (2010). Feature selection for document ranking using best first search and coordinate ascent. *Sigir workshop on feature generation and selection for information retrieval.*

Di, W. & Crawford, M. (2012). View generation for multiview maximum disagreement based active learning for hyperspectral image classification. *Geoscience and Remote Sensing,IEEE Transactions on*, pp. (99):1--13.

Drucker, H.; Burges, C. J.; Kaufman, L.; Smola, A. J. & Vapnik, V. (1997). Support vector regression machines. Em *Advances in neural information processing systems*, pp. 155--161.

Duda, R. O.; Hart, P. E. & Stork, D. G. (2000). *Pattern Classification (2Nd Edition).* Wiley-Interscience. ISBN 0471056693.

Ferreira, R.; Pimentel, M. d. G. C. & Cristo, M. (2015). Exploring graph topology via matrix factorization to improve wikification. Em *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 1099--1104, New York, NY, USA. ACM.

Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Gao, W. & Yang, P. (2014). Democracy is good for ranking: Towards multi-view rank learning and adaptation in web search. Em *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pp. 63--72, New York, NY, USA. ACM.

Geng, X.; Liu, T.-Y.; Qin, T. & Li, H. (2007). Feature selection for ranking. Em *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pp. 407--414, New York, NY, USA. ACM.

Gunning, R. (1952). *The technique of clear writing*. McGraw-Hill, New York.

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157--1182.

Han, J.; Pei, J. & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832--844.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65--70.

Kaufman, L. & Rousseeuw, P. (1987). *Clustering by means of medoids*. North-Holland.

Kleminski, R.; Kajdanowicz, T.; Bartusiak, R. & Kazienko, P. (2017). On quality assesement in wikipedia articles based on markov random fields. Em *Intelligent Information and Database Systems - 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3-5, 2017, Proceedings, Part I*, pp. 782--791.

Kuhn, M. & Johnson, K. (2013). *Applied predictive modeling*. Springer.

Kumar, V. & Minz, S. (2016). Multi-view ensemble learning: An optimal feature set partitioning for high-dimensional data classification. *Knowl. Inf. Syst.*, 49(1):1--59. ISSN 0219-1377.

M. Chen, K. W. & Chen, Y. (2011). Automatic feature decomposition for single view co-training. *In International Conference on Machine Learning.*

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. Em *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 281--297. Oakland, CA, USA.

Mc Laughlin, G. H. (1969). Smog grading-a new readability formula. *Journal of reading*, 12(8):639--646.

Mitchell, T. (1997). Machine learning, mcgraw-hill higher education. *New York.*

Myers, J. L.; Well, A. & Lorch, R. F. (2010). *Research design and statistical analysis.* Routledge.

Oliveira, L. S.; Sabourin, R.; Bortolozzi, F. & Suen, C. Y. (2003). Feature selection for ensembles: A hierarchical multi-objective genetic algorithm approach. Em *Proceedings of the Seventh International Conference on Document Analysis and Recognition-Volume 2*, p. 676. IEEE Computer Society.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825--2830.

Ressler, S. (1993). *Perspectives on electronic publishing: standards, solutions, and more.* Prentice-Hall, Inc.

Senter, R. & Smith, E. A. (1967). Automated readability index. Relatório técnico, DTIC Document.

Shannon, C. E. (2001). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3--55. ISSN 1559-1662.

Sun, S. (2013). A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7):2031--2038. ISSN 1433-3058.

Sun, S.; Jin, F. & Tu, W. (2011). *View Construction for Multi-view Semi-supervised Learning*, pp. 595--601. Springer Berlin Heidelberg, Berlin, Heidelberg.

Witten, I. H.; Frank, E.; Hall, M. A. & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

Zhao, J.; Xie, X.; Xu, X. & Sun, S. (2017). Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43--54.

Zhao, Y.; You, X.; Yu, S.; Xu, C.; Yuan, W.; Jing, X.-Y.; Zhang, T. & Tao, D. (2018). Multi-view manifold learning with locality alignment. *Pattern Recognition*, 78:154 – 166. ISSN 0031-3203.