applied sciences

MDPI

Article

Best Practices of Convolutional Neural Networks for Question Classification

Marco Pota [1,*], Massimo Esposito [1], Giuseppe De Pietro [1] and Hamido Fujita [2,3,4]

1 Institute for High Performance Computing and Networking—National Research Council of Italy (ICAR-CNR), 80131 Naples, Italy; massimo.esposito@icar.cnr.it (M.E.); giuseppe.depietro@icar.cnr.it (G.D.P.)
2 Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City 720000, Vietnam; hfujita-799@acm.org
3 Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18010 Granada, Spain
4 Faculty of Software and Information Science, Iwate Prefectural University, Iwate 020-0693, Japan
* Correspondence: marco.pota@icar.cnr.it

Received: 16 June 2020; Accepted: 3 July 2020; Published: 8 July 2020

Abstract: Question Classification (QC) is of primary importance in question answering systems, since it enables extraction of the correct answer type. State-of-the-art solutions for short text classification obtained remarkable results by Convolutional Neural Networks (CNNs). However, implementing such models requires choices, usually based on subjective experience, or on rare works comparing different settings for general text classification, while peculiar solutions should be individuated for QC task, depending on language and on dataset size. Therefore, this work aims at suggesting best practices for QC using CNNs. Different datasets were employed: (i) A multilingual set of labelled questions to evaluate the dependence of optimal settings on language; (ii) a large, widely used dataset for validation and comparison. Numerous experiments were executed, to perform a multivariate analysis, for evaluating statistical significance and influence on QC performance of all the factors (regarding text representation, architectural characteristics, and learning hyperparameters) and some of their interactions, and for finding the most appropriate strategies for QC. Results show the influence of CNN settings on performance. Optimal settings were found depending on language. Tests on different data validated the optimization performed, and confirmed the transferability of the best settings. Comparisons to configurations suggested by previous works highlight the best classification accuracy by those optimized here. These findings can suggest the best choices to configure a CNN for QC.

Keywords: question classification; multilingual; convolutional neural networks; Natural Language Processing (NLP); deep learning

1. Introduction

Nowadays, intelligent systems able to interact with users in natural language are being developed. However, due to the difficulties associated with natural language understanding by computer systems, this is still a field of research of increasing interest [1–3].

In particular, question answering systems should be able to answer automatically to questions presented in natural language. In order to accomplish this task, a number of operations are required, in order to eventually translating from spoken to written text, to process natural language (tokenization, part-of-speech tagging, dependency parsing), to analyze the question (entity extraction, question classification, query formulation), and to consult the information corpora (information retrieval and answer extraction).

This work concerns the Question Classification (QC) module, which is of primary importance [4,5], since it is in charge of distinguishing different types of questions, corresponding to the expected Lexical Answer Type, enabling the correct extraction of the answer [6].

The function of the QC module is accomplished by a model, trained on a set of already labelled questions. This model classifies textual fragments according to a pre-defined taxonomy. Different types of characteristics, i.e., morphological [7,8], syntactic [4,5], and semantic [9,10], should be considered to interpret text correctly.

However, QC differs from other text classification problems, like sentiment analysis and document categorization, due to the interrogative form and the length that are peculiar of questions. As a consequence, the best performing approach for QC should be chosen peculiarly, and eventually by adopting a different design with respect to methods used for other text classification tasks.

In addition, the ability to classify questions optimally could be accomplished differently, depending on the language [9]. Indeed, syntax rules are different, e.g., English has a more rigid word order to mark the difference between subjects and objects, with respect to Italian. This means that one could have to consider differently sized sets of words together, to catch the same meaning in different languages. Moreover, from the morphological point of view, some languages, like Italian, are richer than others, like English, with impoverished inflection in nouns and verbs; e.g., a single verb in Italian could be inflected in up to 50 different words, while this number is maximum 8 for English. As a consequence, different approaches could be more useful to represent words, depending on the morphological richness of languages.

Various approaches were employed in research literature to tackle short-text classification [11–14] and for QC task in particular [15,16]. However, with respect to classical machine learning approaches, which need the extraction of a big number of features from the text by Natural Language Processing (NLP) methods [17], most recently, great improvements are gained by using neural networks, both from the points of view of the speed of the model and of the classification performance. The text is typically represented by a relatively small number of features obtained by the Word Embedding (WE) process [18], performed by means of a technique chosen among different existing ones. The most common architecture used for text classification by the state-of-the-art solutions was to implement Convolutional Neural Networks (CNNs), since they allow to obtain outstanding results [19,20]. The details of the convolutional architecture and of the learning procedure have been chosen by researchers mainly based on subjective choices. However, as written above, a peculiar solution should be individuated for QC task, probably also depending on the language and on the dataset size.

This work aims at suggesting best practices for using CNNs for QC, with the aim of improving the results of the best existing approaches. However, instead of proposing a different architecture, the basic CNN architecture is implemented with freely adjustable settings, to have insights about their influence and thus choosing the best configuration for the QC problem. Enough numerous experiments, consisting in training and testing the neural network, are executed, to be able to perform a multivariate analysis and evaluate the influence of all the factors and some of their interactions. In particular, words representation, architectural characteristics and hyperparameters, detailed in the following, are all examined as factors, with regard to their potential influence on the QC performance. The expected result consists in the possibility of designing the most performing settings for classifying questions depending on the language.

To the best of our knowledge, with respect to previous similar research [19,21,22], this work is the first one focused on QC, that analyses all the factors involved in the model construction, and their influence on classification performance depending on the language.

More in detail, the main contribution of this work consists in the analysis and optimization of all the factors involved in the CNN design potentially contributing to the improvement of the QC performance and of their interactions:

1. Regarding the text representation, the following approaches are compared here: The inclusion or deletion of punctuation, the use of a well-established pre-trained WE model or of random

vectors for words representation, the use of null vectors or of random vectors for representing Out-Of-Vocabulary (OOV) words, the embedding dimension, and the possibility of fine-tuning WE vectors during learning or keeping them constant;
2. regarding the CNN architecture, which uses filters to extrapolate features relative to sets of consecutive words, the following characteristics are tuned here: Filter region size, number of filters, and the activation function, while only the pooling strategy is fixed; and
3. regarding the learning hyperparameters, for training network weights and eventually the WE vectors, the following are analyzed: Batch size, learning method, learning rate, and regularization terms, while the number of epochs is chosen for each run to ensure convergence.

Moreover, the proposed procedure is performed and settings are tuned for two languages, English and Italian, in order to evaluate differences of the contribution of each factor between languages having different morphological richness, and to demonstrate that a system optimized by the proposed approach can be employed successfully in a multilingual context.

The analysis and the subsequent application of the optimized QC model is performed for two datasets of labelled questions made available in both English and Italian languages by a task presented at Text Retrieval Conferences (TREC) 2002 and 2003 (https://trec.nist.gov/, accessed 1 July 2020). In addition, a widely used dataset of English labelled questions (http://cogcomp.cs.illinois.edu/Data/QA/QC/, accessed 1 July 2020) is employed to check transferability.

Finally, the optimal CNN configurations found here are compared with those found in the most relevant previous similar works, [19,21].

The paper is structured as follows. The following part of this section summarizes related works, while in Section 2 describes the data, formalizes the general QC approach comprising the CNN, and plan the model optimization. The experimental plan with results and their discussion are presented in Section 3. Finally, Section 4 draws conclusions of the work.

Related Works

QC, and more generally speaking sentence classification, is a crucial task for NLP [1,2,16]. Natural language sentences, in both affirmative and interrogative forms, have complicated structures, both sequential and hierarchical, that must be handled to allow their comprehension. Thanks to their ability to capture local relations of temporal or hierarchical structures, CNNs have emerged as a relatively simple yet powerful class of models for sentence modelling and classification, since characterized by remarkably strong performances, with different shallow or deep architectures proposed in the recent years.

The first CNN for sentence classification with end-to-end training is proposed in [23,24]. In this seminal work, one convolutional layer is used together with a new global max-pooling operation, resulting to be very effective for text. Moreover, multiple deep models are co-trained on many tasks to transfer task-specific information. Starting from the results of this work, a simpler architecture with slight modifications have been presented in [21], achieving state-of-the-art performances even on many small datasets. In particular, one convolutional layer with multichannel representation and variable-size filters are employed, where fine-tuned or pre-trained word embeddings are combined in multi-channels, convolutions allow determining high-level abstract features, and multiple linear filters are used to effectively extract different n-gram features. Both the CNN architectures proposed in [23,24] and in [21] make use of max-pooling to keep the most important information to represent the sentence. Moreover, the pooling operation helps the network deal with variable sentence lengths. In [25] a further variant of multi-layer CNN architecture was proposed, with a dynamic k-max-pooling, where k depends on the length of the sentence and can be dynamically set as a part of the network. This allows detecting the k most relevant features occurring into a sentence, independent of their specific position and preserving their relative order. In [26], a multichannel variable-size CNN architecture for sentence classification was described, further exploring the capabilities of multichannel and variable size feature

detectors. In particular, it combines diverse versions of pre-trained word embeddings and extracts features of multi-granular phrases with variable-size convolution filters.

All of CNNs presented in these works are based on word input tokens, encoded as distributed representations in the form of WE vectors [27]. Moreover, they are rather shallow (two layers in most of them), if compared to those successfully proposed to face computer vision problems, due to the reduced length, in terms of number of words, of typical sentences and paragraphs.

Later, a first attempt of CNN jointly using character-level, word-level, and sentence-level representations to perform sentence classification is described in [28], with a shallow architecture made of two convolutional layers to extract relevant features from words and sentences of any size. More recently, a deep CNN architecture, with up to 6 convolutional layers, was proposed in [29], able to automatically learn the notions of words and sentences on texts operating directly at a character level, without any pre-processing, not even tokenization. Convolutional kernels of size 3 and 7 were used, as well as simple max-pooling layers. Another interesting aspect of this work is the usage of several large-scale data sets for text classification. In [30], a new deep CNN architecture, with up to 29 convolutional layers, was proposed for text classification, operating directly at character level and using only small convolutions and pooling operations.

More specifically with respect to QC, in [20], a CNN was used to classify Italian questions. In particular, different solutions regarding the CNN architecture have been tested, and, according to literature advices, the best settings have been searched in the proper ranges, in order to maximize the classification power for the particular case of Italian questions dataset. In [31], an extended CNN architecture is proposed, able to first classify a question into a broader category, and, successively, based on the prior knowledge, assign to it a more specific category. This solution was tested on an English questions dataset with pre-trained word embeddings, showing results on par or improved with respect to other classical methods. In [32], a simple and effective method for QC is presented, which increases generalization, by replacing entities with placeholders, and diversity of sentence features, by reading sentence vectors from both forward and reverse directions. This approach has shown better performance than many other complex CNN models, also proving its effectiveness applied to question answering systems. Finally, in [33], a QC approach based on word embedding using subword information and CNN is outlined, in order to improve classification accuracy. In particular, a comparison between English and Italian languages is reported, by highlighting eventual improvements obtained by initializing word embeddings with advanced vectors learned in an unsupervised manner and comprising character-based information.

Summarizing, all the presented approaches based on CNNs for sentence classification, and specifically for QC, are characterized by models, whose structure is designed by hand by experts, thus requiring considerable skill and experience to select suitable hyperparameters such as the learning rate, the size of convolutional filters, the number of layers and so on. Moreover, these hyperparameters have internal dependencies, which make them particularly expensive for tuning and can depend on the specific classification task considered. Even though some recent works have shown that there exists much room to improve current optimization techniques for learning deep CNN architectures [34], fundamental working principles and behaviors of CNN models when specifically applied to QC have not been extensively investigated.

The most relevant works addressing these issues are generally tested for text classification. In [21], different strategies for words representation are compared, by employing in turn, singularly or combined in a multi-channel way, differently initialized, and eventually fine-tuned WE vectors. On the basis of [21] model, a sensitivity analysis of CNNs is proposed in [19], summarizing the influences of various hyperparameters, i.e., WE vectors, filter size, number of filters, activation function, pooling strategy, and regularization. Both these works, for the QC task considered among the others, found different best settings with respect to the other tasks.

However, to the best of our knowledge, related research considered only few settings, and without reference to possible interactions among them. Moreover, (i) QC was only considered as an instance of text classification; (ii) the possible relation with the language not taken into account.

Thus, this work constitutes the first attempt of considering hyperparameters in a comprehensive way, examining different possibilities with respect to morphologically different languages, to study the problem of configuring the appropriate CNN architecture for QC.

2. Materials and Methods

2.1. Data

QC aims at associating each question to a class comprised in a given set. This is made accordingly to a number of examples of labelled questions, used to train and test the model.

Different datasets are available, particularly for English language. The main example is the TREC dataset provided by [35], used in various previous works, comprising [21], which is particularly big. However, in order to study questions in different languages, multilingual data are rare and less extensive.

In order to compare English and Italian languages, the chosen data are made of the union of two datasets presented at TREC conferences 2002 and 2003, each comprising 500 training questions and labelled according to the same taxonomy. The same 1000 questions are available in English and Italian, among the other languages. For example, a row of the joint dataset is made of four attributes (coarse class, fine class, question in English, question in Italian), as follows: "FACTOID—LOCATION—What is Africa's largest country?—Qual è il paese più vasto dell'Africa?".

In Table 1, the two-levels taxonomy is reported.

Table 1. Taxonomy of question classes.

Coarse Classes	Fine Classes
Definition	Location, Person, Other.
Factoid	Acronym, How, Location, Material, Measure, Person, Time, Title, Other.
List	Location, Person, Title, Other.

Since the aim of the approach is the single (not hierarchical) classification task, coarse classes were not considered in this work. On the other hand, all the questions were included, and the union of fine classes for any of the three coarse classes is considered, which results in the following 9 labels: "Acronym", "How", "Location", "Material", "Measure", "Person", "Time", "Title", "Other".

Each experiment is performed by 10-fold cross-validation. Therefore, the runs are performed with a number of examples for training $Ntrain = 900$, and a number of examples for testing $Ntest = 100$.

Moreover, the dataset provided by (Li and Roth 2002), available online, is also used, to compare results with those of other state-of-the-art best convolutional architectures. It is already divided into 5452 questions for training and 500 for testing, and is based on a 2-levels taxonomy, whose coarse level, used here, is made of the following 6 classes: "ABBREVIATION", "ENTITY", "DESCRIPTION", "HUMAN", "LOCATION", "NUMERIC".

2.2. Question Classification Model

This section describes the structure of the model employed for classifying questions, and the learning procedure. This model, firstly developed in [23,24], was implemented with variable settings in the open source Python framework TensorFlow (https://www.tensorflow.org/, accessed 1 July 2020). The testing platform consisted of a fold containing data, a main program with subroutines for pre-processing and model architecture and producing the results, and three configuration json files, where the user can manually change all the settings before each run. The variable settings were defined within sets chosen coherently with findings of previous literature and with preliminary experiments. The model general form

is schematized in Figure 1. It comprised a pre-processing phase, which allowed translation of the question into a sparse matrix constituting the input layer, the embedding phase, which allowed representation of the question by a matrix with smaller dimension constituting the embedding layer, and a CNN made of convolutional layer, pooling layer, fully connected layer, and output layer, which finally associated each question to a class.

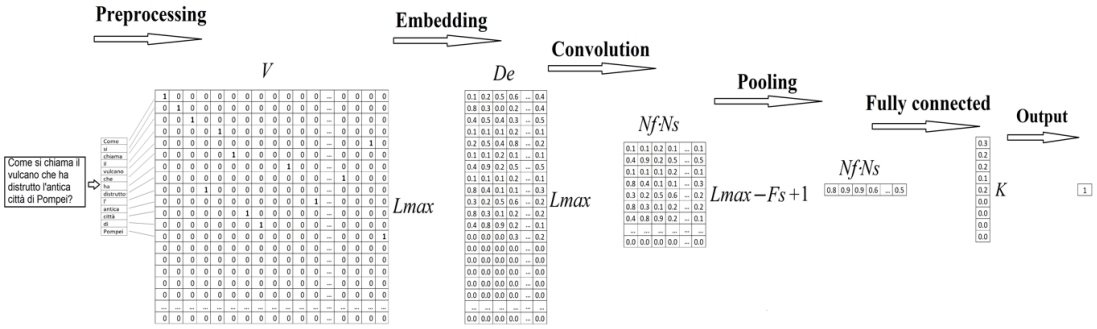

Figure 1. The Question Classification model.

The following subsections describe in detail the pre-processing and WE phases (Section 2.2.1), the CNN architecture (Section 2.2.2), and the procedure for learning network weights (Section 2.2.3). Finally, since this approach is implemented here with freely adjustable settings, Section 2.2.4 summarizes the degrees of freedom considered with respective possible values, and explains the optimization approach.

2.2.1. Question Pre-Processing and Word Embedding

Each question had to be pre-processed, to be divided into a sequence of tokens, and represented as a sparse matrix.

Firstly, special characters, not comprised in the set {A-Za-z0-9(),;.:!?’’”} were substituted with spaces. Then, apostrophes and some substrings comprising them were substituted, depending on the language, as reported in Table 2.

Table 2. Substitution of strings comprising apostrophes.

English	
Original Characters	**Substitution**
“ (double typewriter apostrophe)	“ (quotation marks)
‘‘ (double backtick)	“ (quotation marks)
‘ve (apostrophe+“ve”)	have (space+“have”)
n’t (“n”+apostrophe+“t”)	not (space+“not”)
‘re (apostrophe+“re”)	are (space+“are”)
‘s (apostrophe+“s”)	‘s (space added before)
‘d (apostrophe+“d”)	‘d (space added before)
‘ll (apostrophe+“ll”)	‘ll (space added before)
Italian	
Original Characters	**Substitution**
“ (double typewriter apostrophe)	“ (quotation marks)
‘‘ (double backtick)	“ (quotation marks)
‘a (apostrophe+“a”)	‘ a (space added between)
‘e (apostrophe+“e”)	‘ e (space added between)
‘i (apostrophe+“i”)	‘ i (space added between)
‘o (apostrophe+“o”)	‘ o (space added between)
‘u (apostrophe+“u”)	‘ u (space added between)
a’ (“a”+apostrophe)	à (“a” with grave accent)
e’ (“e”+apostrophe)	è (“e” with grave accent)
i’ (“i”+apostrophe)	ì (“i” with grave accent)
o’ (“o”+apostrophe)	ò (“o” with grave accent)
u’ (“u”+apostrophe)	ù (“u” with grave accent)

The possibility of eliminating the other standard punctuation symbols {(),;.:!?‘’"} is a degree of freedom:

$$AvoidPunctuation = \begin{cases} \text{True} \\ \text{False} \end{cases}. \quad (1)$$

Therefore, if they were eliminated (*AvoidPunctuation* = True), they were substituted with spaces, otherwise (*AvoidPunctuation* = False) a space was added before and after each of them. Finally, sets of consecutive spaces were substituted with only one space. At this point, the text was already divided in tokens by spaces.

Each question was made of L tokens, and the maximum length *Lmax* was calculated over the whole dataset. Moreover, a vocabulary was assembled by gathering all V different tokens plus an entry <UNK> in the first position corresponding to unknown token. Original tokens are used, instead of lemmatizing them, to be coherent with pre-trained WE.

Once the vocabulary was fixed, each token was represented as a vector with V elements, which were all equal to 0, except the element corresponding to the position of the token in the vocabulary, equal to 1. Therefore, each question was represented as a matrix $\mathbf{X}$ with V columns and *Lmax* rows, composed by vectors $\mathbf{x}_j$, with $j = 1, \ldots, Lmax$, where if $L < Lmax$, last rows were filled with all zeros. This matrix was the input layer of the deep neural network.

The next embedding phase consisted in the linear transformation of $\mathbf{X}$ into a matrix with smaller dimension. Each one-hot V-dimensional vector $\mathbf{x}_j$ was transformed into a De-dimensional vector corresponding to the representation of the word suggested by the pre-trained WE model or to a random or null vector. In practice, $\mathbf{X}$ was multiplied by the embedding matrix $\mathbf{W_{emb}}$ with De columns and V rows, to obtain a matrix $\mathbf{X_{emb}}$ made of De columns and *Lmax* rows:

$$\mathbf{X_{emb}} = \mathbf{X}\mathbf{W_{emb}}. \quad (2)$$

The embedding matrix was initialized depending on the choice of the *WEinit* factor:

$$WEinit = \begin{cases} \text{pre} - \text{trained} \\ \text{random} \end{cases}. \quad (3)$$

If pre-trained WE vectors are used (*WEinit* = pre-trained), then the row of $\mathbf{W_{emb}}$ corresponding to each known word was initialized as the pre-trained WE vector, while the other rows corresponding to OOV words were initialized with null vectors (*OOVinit* = null). The pre-trained WE representation chosen for this work was based on fastText model, with 300 dimensions (De = 300), trained on the Wikipedia corpora (https://fb-public.app.box.com/s/htfdbrvycvroebv9ecaezaztocbcnsdn, accessed 1 July 2020), both in English and Italian languages. This model was chosen for its outstanding characteristics. In fact, it was an evolution of the skip-gram model, which trains the representation of each word by unsupervised learning to predict words that appear in its context, but fastText also measures similarity between words based on character n-grams included in them. Therefore, these vectors encode information regarding syntactic structure of the text and semantic features like the skip-gram model, as well as information regarding the morphology of the words.

On the other hand, if pre-trained vectors were not used (*WEinit* = random), then all the rows of $\mathbf{W_{emb}}$ were initialized with random vectors, both for known or unknown words (*OOVinit* = random). This representation was made with a number of dimension which was a further degree of freedom, studied in the following interval:

$$De \in [10, 500]. \quad (4)$$

Since the values assumed by *OOVinit* are coupled with those assumed by *WEinit*, in the following *OOVinit* was omitted.

In both cases, the embedding matrix $\mathbf{W_{emb}}$ could be kept constant or fine-tuned during the network training:

$$WEtuning = \begin{cases} \text{static} \\ \text{dynamic} \end{cases}. \tag{5}$$

2.2.2. Convolutional Neural Network Architecture

A classical CNN architecture was used here for associating questions with labels. However, here the architecture was not fixed, but was implemented with freely adjustable settings.

A convolution was firstly applied to $\mathbf{X_{emb}}$, by using a single channel, with no padding and stride 1, as recommendable in text classification context. Filters of different sizes may be employed, therefore, if there were *Ns* different sizes and for each size a number *Nf* of filters, the total number *Ntot* of filters was:

$$Ntot = Nf \cdot Ns. \tag{6}$$

The sizes and the total number of filters were degrees of freedom, and they were considered in the following ranges:

$$Fs \in [1, 10], \tag{7}$$

$$Ntot \in [50, 500]. \tag{8}$$

Each filter of a certain size consists in a matrix $\mathbf{W^i_{conv}}$, with $i = 1, \ldots, Ntot$ made of *De* columns and *Fs* rows. The result of the convolution was a vector $\mathbf{x_{conv}}^i$ with dimension $Lmax - Fs + 1$, whose components $x^{i,j}_{\text{conv}}$, with $j = 1, \ldots, Lmax - Fs + 1$, can be written as:

$$x^{i,j}_{\text{conv}} = \sum_{jj=1}^{Fs} \sum_{d=1}^{De} \left(\mathbf{X_{emb}}[j + jj - 1][d] \cdot \mathbf{W^i_{conv}}[jj][d] \right). \tag{9}$$

Then a bias term b^i_{conv} was added to each component, and an activation function f was applied, to get each component $x^{i,j}_{\text{act}}$, with $j = 1, \ldots, Lmax - Fs + 1$, of the vector $\mathbf{x^i_{act}}$, which was the final result of the convolution by the given filter $\mathbf{W^i_{conv}}$:

$$x^{i,j}_{\text{act}} = f\left(x^{i,j}_{\text{conv}} + b^i_{\text{conv}} \right). \tag{10}$$

Of course, vectors $\mathbf{x^i_{act}}$ with the same size were obtained by using filters with the same *Fs*, while vectors of different sizes were obtained by differently sized filters. However, *Ntot* vectors were obtained, and they constitute the convolutional layer.

The activation function to use for convolution was a degree of freedom of the proposed implementation. The following functions were used, whose meaning is shown in Figure 2:

$$f = \begin{cases} \text{eLU} \\ \text{Identity} \\ \text{ReLU} \\ \text{sigmoid} \\ \text{softplus} \\ \text{softsign} \\ \tan \text{h} \end{cases}. \tag{11}$$

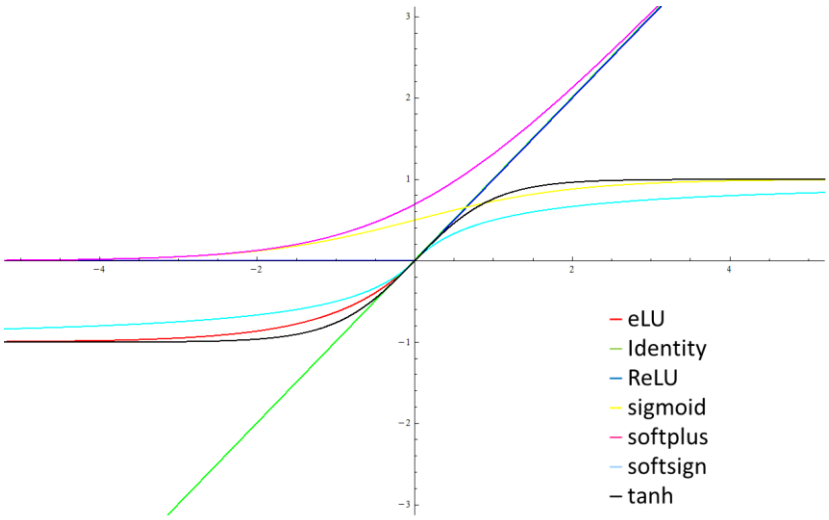

Figure 2. The activation functions employed.

The following operation was the pooling, which was implemented coherently with the common choice, i.e., the 1-max pooling strategy. In fact, using the max function was forced by the padding with zeros the input representation of questions shorter than *Lmax*, and the choice of only one maximum element was certainly enough, due to the big number of filters employed. Therefore, the pooling layer was constituted by a horizontal vector $\mathbf{p}$ with dimension *Ntot*, whose elements p^i, with $i = 1, \ldots, Ntot$, were:

$$p^i = \max \mathbf{x}^i_{\text{act}}. \tag{12}$$

The following fully connected layer was constituted by K neurons, where K was also the number of classes. In the considered case, $K = 9$. The vector of class activations $\mathbf{y}$ was computed by multiplying $\mathbf{p}$ by a matrix of weights $\mathbf{W_{fc}}$ with *Ntot* rows and K columns, and adding a bias vector $\mathbf{b_{fc}}$:

$$\mathbf{y} = \mathbf{p}\mathbf{W_{fc}} + \mathbf{b_{fc}}. \tag{13}$$

The final output layer was made of only one node, which contains the position of the class with the highest activation:

$$output = \text{argmax}\mathbf{y}. \tag{14}$$

2.2.3. Learning Procedure

The described model includes many parameters that were initialized randomly and have to be trained, i.e., $\mathbf{W^i_{conv}}$ with $i = 1, \ldots, Ntot$, b^i_{conv} with $i = 1, \ldots, Ntot$, $\mathbf{W_{fc}}$, and $\mathbf{b_{fc}}$, for a total of $De \cdot Fs \cdot Ntot + Ntot \cdot K + K$. Moreover, $\mathbf{W_{emb}}$ can be initialized by pre-trained WE or randomly, but in both cases they were fine-tuned if *WEtuning* = Dynamic, bringing other $V \cdot De$ parameters.

These parameters were adapted on data by a learning procedure summarized as follows.

Firstly, the training dataset was divided in batches composed of a certain number of examples. In this work, the batch size was a degree of freedom, studied in its whole range:

$$batch \in [1, 900]. \tag{15}$$

All examples of a batch were used as input of the model, but during the training, in order to learn separately different parts of the network, the pooling layer was modified by the dropout function, which randomly transforms each component p^i multiplying it by zero with probability $(1 - P_{keep})$,

and by $1/P_{keep}$ with probability P_{keep}, so that the expected sum remains unchanged. Here, the dropout was a degree of freedom, variable in the following interval:

$$P_{keep} \in (0.0, 1.0]. \tag{16}$$

For each input b, the loss was calculated by the cross entropy function (19), where p_k (17) was the softmax transform of the kth component of computed vector $\mathbf{y}$, and c_k (18) was 1 for the position of the true label k_{True}, 0 otherwise:

$$p_k = \frac{e^{y_k}}{\sum_{\kappa=1}^{K} e^{y_\kappa}}. \tag{17}$$

$$c_k = \begin{cases} 1 \text{ if } k = k_{\text{True}} \\ 0 \text{ otherwise} \end{cases}. \tag{18}$$

$$loss_b = -\sum_{k=1}^{K} c_k \cdot \log(p_k). \tag{19}$$

After a batch, the whole associated loss was calculated as:

$$loss = \frac{1}{batch} \sum_{b=1}^{batch} loss_b + l2 \cdot \left(\frac{|\mathbf{W_{fc}}|^2}{2} + \frac{|\mathbf{b_{fc}}|^2}{2} \right). \tag{20}$$

The regularization parameter $l2$, used to prevent big values of fully connected layer weights, was a degree of freedom here, studied in the following interval:

$$l2 \in [1.0, 5.0]. \tag{21}$$

The loss gradient was used for updating network weights by a backpropagation approach based on Stochastic Gradient Descent (SGD) algorithm, which implies a stochastic approximation [36] of the basic gradient descent algorithm. Since it reduces the computational complexity, achieving faster iterations in trade for a lower convergence rate [37], it was recognized as a very effective learning algorithm in machine learning [38]. A variant of the updating rule was freely chosen among the following ones:

$$optimizer = \begin{cases} \text{Adadelta} \\ \text{Adagrad} \\ \text{Adam} \\ \text{Ftrl} \\ \text{GradientDescent} \\ \text{Momentum} \\ \text{ProximalAdagrad} \\ \text{ProximalGradientDescent} \\ \text{RMSProp} \end{cases}. \tag{22}$$

While in case *optimizer* = Momentum, the momentum parameter was fixed to 0.1, according to previous findings, the learning rate, which was a further parameter common to all the algorithms, was the last considered degree of freedom, studied in the following wide range, enlarged with respect to previous works [19,21]:

$$\eta \in [0.01, 10]. \tag{23}$$

The intent in this work was to get the best possible model, therefore the number of epochs was not taken as an adjustable setting.

2.2.4. Threats to Validate

Different threats could affect the learning procedure.

First, underfitting could affect results, if the number of epochs chosen for learning was too low. In order to avoid it, a sufficient number of epochs was chosen for different runs. For most of the runs, 2000 epochs result enough to reach convergence, while in some cases (properly compared with the others) 20000 epochs were needed.

Second, a large number of epochs could cause overfitting on training data. In order to avoid it, every 2 epochs, the model was tested on a randomly sampled dev set, and at the end of the epochs, the model presenting the best accuracy on the dev set was chosen.

Finally, the choice of training and testing questions within the dataset could (positively or negatively) influence and distort the results. In order to avoid it, each experiment was performed by stratified 10-fold cross-validation. Therefore, the dataset was randomly divided into 10 subsets of 100 questions with approximately the same rate of labels. Each run was performed with the union of 9 question subsets for training, and the remaining subset for testing, this was repeated 10 times for considering all the examples for testing, and the results of the 10 runs were averaged to obtain the result of the experiment.

2.2.5. Model Optimization

In order to optimize the QC model, its classification accuracy was studied by analyzing different experiments, corresponding to respective configurations of settings (factors).

Since each experiment was the set of 10 training and testing phases constituting a 10-fold cross-validation, the accuracy of an experiment was the average of the accuracies gained by the 10 trained models on the respective test set. The accuracy on a test set was calculated as a percentage, by averaging c_{output}, which was 1 if the network output (14) was equal to the position of the true label associated with the bth input of the test set ($output = k_{\text{True}}$), 0 otherwise:

$$Acc = \frac{1}{Ntest} \sum_{b=1}^{Ntest} c_{output} \cdot 100\%. \tag{24}$$

The factors here considered to analyze their influence on the model accuracy are summarized in Table 3. For categorical factors, all the possible values were considered, while for quantitative ones the considered admitted ranges were based on previous literature findings.

Since considering all the possible interactions among factors would involve an unfeasible experimentation, some factors were analyzed in the following one by one, since they were hypothesized to have negligible interactions with the others, while some sets of factors were studied together to verify potential interactions.

For each factor or set of factors, their individual influences and interactions (effects) were evaluated in a chosen range by performing a set of experiments. Most sets of experiments were planned according to full factorial designs, which comprise all the combinations of factors levels. This approach needs more numerous experiments, but minimizes the risk of confounding different effects. The range of each factor, and the fixed values of other settings, relative to factors not being evaluated in a set of experiments, since were hypothesized to not interact, were chosen according to findings of previous works [19–21,33], or to preliminary experiments.

Table 3. Freely adjustable settings of the Question Classification (QC) model analyzed here and their admitted values.

Setting	Symbol	Set of Admitted Values
Text representation		
Eliminate punctuation	*AvoidPunctuation*	{True,False}
Pre-trained WE vectors for known words and null vectors for OOV words, or random vectors for all the words	*WEinit*	{pre-trained,random}
Embedding dimension	*De*	[10,500]
Fine tuning of WE vectors together with other network weights during training	*WEtuning*	{static,dynamic}
CNN architecture		
Filter size	*Fs*	[1,10]
Total number of filters	*Ntot*	[50,500]
Activation function	*f*	{eLU,Identity,ReLU,sigmoid,softplus,softsign,tanh}
Learning procedure		
Batch size	*batch*	[1,900]
Probability that dropout function keeps a node	P_{keep}	(0.0,1.0]
Parameter of loss regularization	*l2*	[1.0,5.0]
Weights updating rule	*optimizer*	{Adadelta,Adagrad,Adam,Ftrl,GradientDescent,Momentum, ProximalAdagrad,ProximalGradientDescent,RMSProp}
Learning rate	η	[0.01,10]

Due to the random initialization of weights and to some other sources of randomness in the learning procedure (splitting training data in batches, dropout function, and SGD algorithm), each run, and thus each whole experiment, gave different results if repeated. Therefore, some repetitions were performed, to estimate the experimental variance σ2, which was used to evaluate the experiments reproducibility.

The intrinsic variance in the measurement of the experiment performance implies that a deterministic functional dependence between factors and model accuracy does not exist. Therefore, in order to analyze the effects on the QC accuracy, an approximate function was extrapolated from each set of experimental results:

$$acc = c_0 + c_1x_1 + c_2x_2 + c_{12}x_1x_2 + \dots \quad (25)$$

where x_1,x_2, … represent the individual factors, x_1x_2, … represent their interactions, and coefficients c_0,c_1,c_2,c_{12}, … were used to linearly combine these (also nonlinear) effects to predict experimental accuracy.

The significance of effects was evaluated in terms of the respective coefficients [39]. Indeed, each estimated coefficient belongs to a respective confidence interval, corresponding to the interval comprising the true coefficient value with 95% probability, that was calculated as follows. Given the estimated experimental variance σ^2, calculated with a certain number of degrees of freedom, the variance of each coefficient can be estimated as σ^2/N, where N is the number of experiments of the full factorial design. Therefore, the width of the coefficient confidence interval can be calculated as $\sigma/N^{1/2} \cdot t_{0.975}$, where $t_{0.975}$ is the value of a t-student distribution with the same degrees of freedom corresponding to 0.975 cumulative probability (two tails t-test). As a consequence, if the estimated coefficient was lower than the confidence interval semi-width, then the confidence interval comprises the null value, and the hypothesis that the true coefficient value was zero cannot be rejected, and the corresponding effect was not significant.

Moreover, the estimated function (24) comprising significant effects can be used to predict the accuracy, on the basis of the considered factors and their eventual interactions. This allows finding optimal values of factors, corresponding to higher calculated accuracy.

After that all the factors were individually optimized, some repetitions corresponding to the optimal settings were performed, to evaluate the performance of the QC model in optimal conditions. Moreover, optimal conditions were validated on a larger set of data.

3. Results and Discussion

In this section, the results obtained by the QC model are reported and discussed. As described before, the accuracy reported in correspondence of a configuration of settings was obtained by the average of 10 cross-validation runs.

The results were obtained for each configuration by considering questions in both English and Italian languages.

First of all (Section 3.1), repeated experiments using the same configuration are described and discussed. Then, the influence of settings regarding text representation (Section 3.2), network architecture (Section 3.3), and learning procedure (Section 3.4) was evaluated. In addition, Section 3.5 takes into account all the previous findings to individuate the most influencing parameters. Finally, Section 3.6 presents optimal settings obtained for different cases, and evaluates the performance of the associated proposed models, also showing a comparison with baseline models found as optimal in previous literature, on a widely used dataset.

In total, 2404 runs for training and testing the described QC model were performed.

3.1. Repetitions

As explained before, each experiment reported here was made of 10 runs since cross-validation is performed. Therefore, for each experiment, an "internal" variance of the testing accuracy is calculated. Averaging on all the experiments, the "internal" standard deviations found were about 4.0% for English and 3.5% for Italian. These quite low values were due to the robustness of the random stratified splitting of the dataset in folds.

On the other hand, some whole experiments were performed 5 times, to evaluate their reproducibility. In the hypothesis that the system is homoscedastic, the accuracy variance could be estimated in correspondence of only one configuration. Here, this hypothesis is relaxed, due to the structural differences between runs performed by using fixed pre-trained WE vectors or random vectors, and between runs performed with fixed WE vectors or by fine-tuning them. Therefore, the accuracy variance is estimated in correspondence of the combinations of these settings. For each configuration, 5 repetitions of the same experiment were performed. In Table 4, the experimental variance σ^2 calculated over repetitions is reported.

Table 4. Variance of testing accuracy over experiments repetitions [1].

		*WEinit**WEtuning*	**Static**	**Dynamic**
σ^2	**English**	**random**	0.37	0.86
		pre-trained	0.14	0.23
	Italian	**random**	0.32	0.07
		pre-trained	0.72	0.13

[1] Other settings were: *AvoidPunctuation* = True, *De* = 300, *Fs* = {1,2,3}, *Ntot* = 300, *f* = ReLU, *batch* = 900, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

Since the estimated variance is itself a random variable, on the basis of results of Table 4, the homoscedasticity can be hypothesized, also with respect to the language; therefore, the experimental variance was estimated by averaging over different configurations the mean pure squared errors of the repetitions. It corresponds to a low standard deviation σ = 0.6% (calculated with 32 degrees of freedom, therefore $t_{0.975}$ = 2.038), and compared to the variance of different experiments, corresponds to a good reproducibility (= 0.90).

3.2. Text Representation

The first setting analyzed here regards the text representation, and in particular, the possibility of eliminating all punctuation symbols from the question during pre-processing. This qualitative

factor can assume 2 levels, and it was hypothesized to not interact with others. Therefore, the only 2 experiments for each language reported in Table 5 were performed.

Table 5. Testing accuracy obtained by eliminating or not punctuation from text [1].

AvoidPunctuation		True	False
Acc (%)	**English**	88.1	88.3
	Italian	86.6	86.8

[1] Other settings were: *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, f = ReLU, *batch* = 10, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

Even if for both languages the case *AvoidPunctuation* = True gives slightly higher accuracy, the differences with the case *AvoidPunctuation* = False were statistically not significant, since it is comparable to (and even smaller than) the standard deviation of repeated experiments. In other words, if a model describing the accuracy as a function of this variable is constructed, the linear coefficient results 0.1% for both languages, which is smaller than the semi-width of its confidence interval (0.9%), therefore the chance that the true value of the coefficient is zero cannot be discarded.

This finding suggests that, in order to simplify the QC model, punctuation can be eliminated without significant loss of information.

Other two factors regarding the text representation, i.e., the possibility of initializing WE vectors of known words by fastText pre-trained vectors and OOV words by null vectors or initializing all WE vectors randomly, and the possibility of fine-tuning these vectors during training or not, were analyzed together, to evaluate at the same time their effects and eventual interactions. Each of these qualitative factors can assume 2 levels; therefore, 4 configurations for each language were tested. For each configuration, 5 repeated experiments were performed, used to estimate variances reported in Table 4, and whose mean accuracies are reported in Table 6.

Table 6. Testing accuracy obtained by different Word Embedding (WE) initialization and fine-tuning strategy [1].

		*WEinit**WEtuning*	Static	Dynamic
Acc (%)	**English**	**random**	77.6	77.8
		pre-trained	77.6	80.2
	Italian	**random**	76.1	76.1
		pre-trained	75.2	80.4

[1] Other settings were: *AvoidPunctuation* = True, *De* = 300, *Fs* = {1,2,3}, *Ntot* = 300, f = ReLU, *batch* = 900, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

In this case, the effects of the evaluated factors *WEinit* and *WEtuning*, and of their interaction, produced significant results. In particular, random or pre-trained WE vectors give equivalent results if they are static, and fine-tuning of random vectors does not improve accuracy, but the combination of *WEinit* = pre-trained and *WEtuning* = dynamic gives a contribution to the mean accuracy of about 2.6% for English and 4.3% for Italian. These contributions are greater than the confidence interval semi-width (about 0.6%).

This behavior can be explained by observing that the generally valid information embodied by WE pre-trained vectors was not necessarily the same required by the specific classification task, therefore, if kept static, they could result equivalent to random ones; however, they embody semantic information that allows, if properly fine-tuned, to get closer to optimal, with respect to random ones.

These findings suggest the following considerations:

- It is convenient to employ fastText pre-trained vectors to initialize WE vectors, which embody semantic and morphological information in words representation;

- it is convenient to fine-tune WE vectors, since optimizing the representation of the single words most influencing on QC allows to stress their importance;
- a significant improvement on QC accuracy is gained if these two settings were used at the same time, since the WE vectors of words semantically associated with question classes, already represented by embodying semantic information, can be coherently optimized; and
- all these effects result more relevantly in the Italian language, with respect to English, since all the improvements regarding words representation were more useful for a morphologically rich language.

The last factor taken into account for text representation was the embedding dimension *De*, hypothesized to have no interactions with the others. This quantitative factor was analyzed in the range [10,500], and in particular, in correspondence of the representative levels {10,100,300,500}, to analyze also its nonlinear effects. Therefore, four configurations for each language were tested, and results are reported in Table 7 and shown in Figure 3.

Table 7. Testing accuracy obtained by different embedding dimensions [1].

De		10	100	300	500
Acc (%)	**English**	54.8	74.0	77.8	78.1
	Italian	41.0	73.7	76.1	75.7

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = random, *WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, f = ReLU, *batch* = 900, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

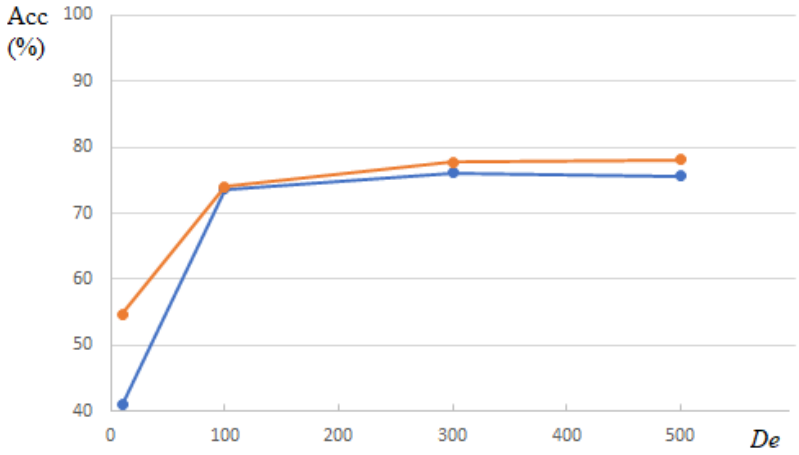

Figure 3. Testing accuracy obtained by different embedding dimensions, for English (**red**) and Italian (**blue**).

From results reported in Table 7, a function was fitted for each language to predict accuracy as a quadratic function of *De* in logarithmic scale, and all coefficients result much greater than their confidence interval, therefore, *De* gives significant effects. The fitted function, in accordance with results shown in Figure 3, explains that, as the embedding dimension increases, a great improvement to QC was given, since more semantic, syntactic, and morphological aspects of words were represented. However, for more than some hundreds dimensions, a plateau was reached, and adding other dimensions does not give a significant improvement.

Therefore, also in accordance with most of the previous literature works, and with the majority of the available pre-trained WE vectors, the value *De* = 300 was chosen as optimal here.

3.3. CNN Architecture

The CNN architecture was analyzed firstly in terms of both the filter size (7) and their total number (8), and then with regard to the activation function involved in (10), while their interactions were neglected.

Different filter sizes were experimented in the range [1,10], taking into account all possible sizes. The functions fitted in this whole range to predict accuracy reveal no significant linear or quadratic effect; however, in the restrictions of this range, the experimental results showed significant trends, associated with significant improvements in correspondence of individual filter sizes with respect to the others, as discussed in the following.

Firstly, all filters with the same size were employed. Results for both languages are reported in Table 8 and shown in Figure 4.

Table 8. Testing accuracy obtained by different filter sizes [1].

Fs		1	2	3	4	5	6	7	8	9	10
Acc (%)	**English**	79.0	81.8	79.4	79.2	78.9	78.3	78.6	79.7	78.3	78.0
	Italian	78.6	81.0	79.3	78.3	79.3	77.6	78.1	76.5	77.2	77.5

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Ntot* = 300, *f* = ReLU, *batch* = 900, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

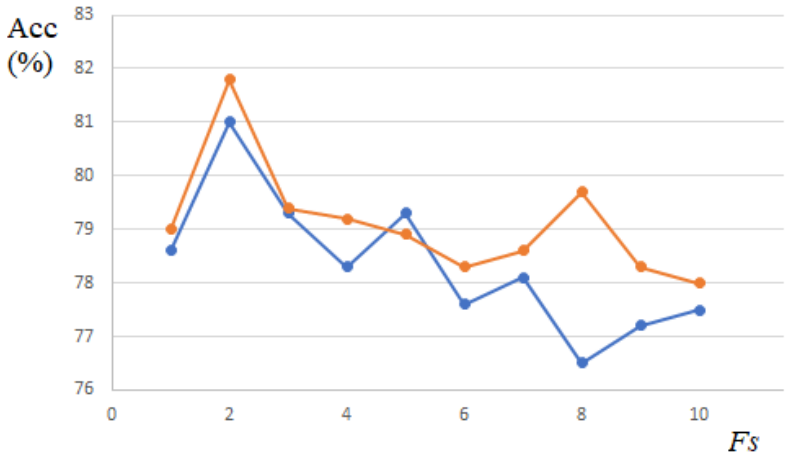

Figure 4. Testing accuracy obtained by different filter sizes, for English (**red**) and Italian (**blue**).

Within the first set of experiments, the best single filter size results *Fs* = 2, which corresponds to significant improvements with respect to both *Fs* = 1 and *Fs* > 2. The trend was similar for both languages, while a misalignment results for *Fs* = 8, which may be due to experimental variance.

Then, in order to evaluate the possibility of using filters of different sizes at the same time, as suggested by previous works [19,20], 150 filters of size 2 were fixed, while the size of the other 150 was varied in the same interval. Results for both languages are reported in Table 9 and shown in Figure 5.

Table 9. Testing accuracy obtained by different filter sizes [1].

Fs		{2,1}	{2,2}	{2,3}	{2,4}	{2,5}	{2,6}	{2,7}	{2,8}	{2,9}	{2,10}
Acc (%)	**English**	79.9	81.8	80.9	79.8	81.2	80.8	80.4	80.1	79.9	80.7
	Italian	79.4	81.0	80.6	80.1	79.2	79.0	80.3	78.9	78.3	79.4

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Ntot* = 300, *f* = ReLU, *batch* = 900, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

These results again allow to individuate *Fs* = 2 as the best filter size, also in association with other filters of size 2. This corresponds to significant improvements with respect to *Fs* = 1 and slight improvements with respect to *Fs* > 2.

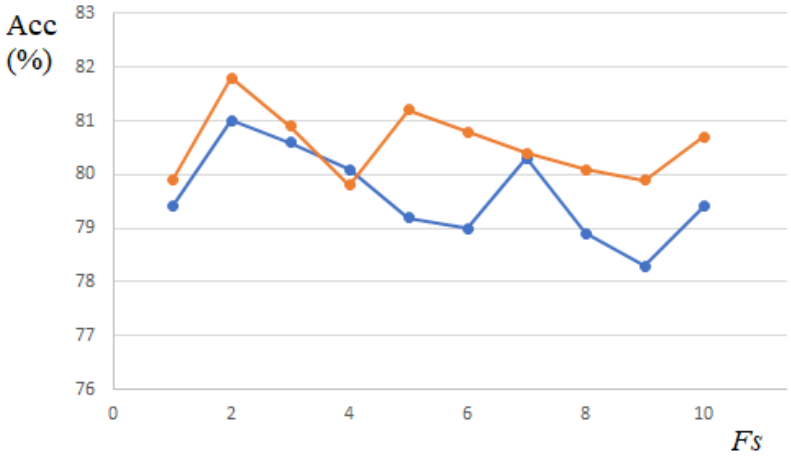

Figure 5. Testing accuracy obtained by different filter sizes apart from 150 filters of size 2, for English (**red**) and Italian (**blue**).

These results mean that substituting 150 filters with others having different sizes does not improve the accuracy. Therefore, further experiments were performed by fixing 200 filters of size 2, while the size of only 100 varies. Results for both languages are reported in Table 10 and shown in Figure 6.

Table 10. Testing accuracy obtained by different filter sizes [1].

Fs		{2,2,1}	{2,2,2}	{2,2,3}	{2,2,4}	{2,2,5}	{2,2,6}	{2,2,7}	{2,2,8}	{2,2,9}	{2,2,10}
Acc (%)	**English**	80.2	81.8	81.4	81.5	80.4	81.0	80.8	80.8	80.8	80.6
	Italian	79.2	81.0	80.4	79.9	80.3	80.5	79.6	79.6	79.3	80.3

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Ntot* = 300, f = ReLU, *batch* = 900, P_{keep} = 0.5, $l2$ = 3.0, *optimizer* = Adadelta, η = 0.1.

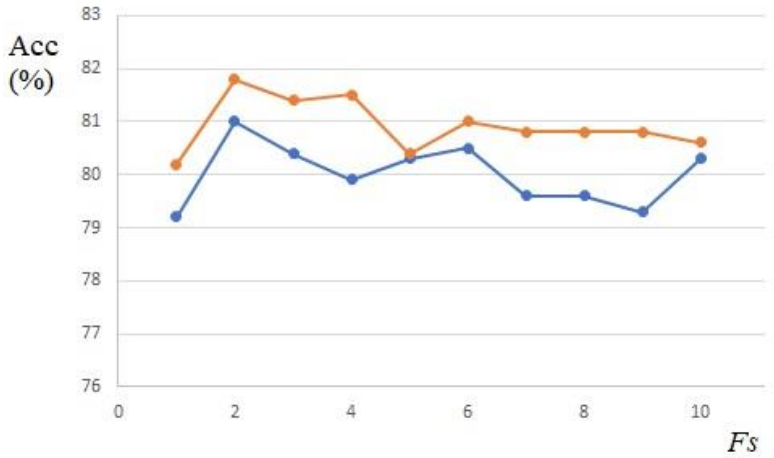

Figure 6. Testing accuracy obtained by different filter sizes apart from 200 filters of size 2, for English (**red**) and Italian (**blue**).

Also in this case, the filter size 2 results the best. However, it corresponds to significant improvements with respect to $Fs = 1$, while the variations for $Fs \geq 2$ were not significant, since they were comparable with the confidence interval of the linear coefficient of the function approximating this trend (about 0.9%).

This finding of the best filter size corresponding to $Fs = 2$ can be explained by observing that, while other literature results were inferred for classifying sentences, if questions were considered as in this work, their classification can be done for most of them by considering a sequence of maximum 2 words

comprised in them. This is coherent with previous findings, e.g., [4], individuating single words like "head words", or "WH-words" (*why*, *when*, *where*, ...), or couples of words (*how much*, *how long*, ...) as the most informative for QC.

The total number of filters was analyzed as well, within the range [50,500], by considering the following values: {50,100,200,300,400,500}. Results are reported in Table 11 and shown in Figure 7.

Table 11. Testing accuracy obtained by different numbers of filters [1].

Ntot		50	100	200	300	400	500
Acc (%)	**English**	80.8	81.7	81.5	81.8	81.0	81.3
	Italian	80.5	81.4	81.1	81.0	80.5	79.4

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Fs* = 2, *f* = ReLU, *batch* = 900, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

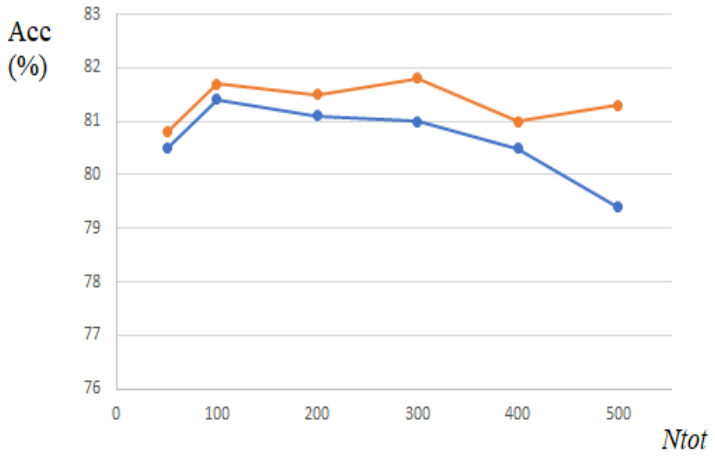

Figure 7. Testing accuracy obtained by different numbers of filters, for English (**red**) and Italian (**blue**).

Also in this case, in the whole considered range, the functions fitted to predict accuracy reveal no significant linear or quadratic effect, but a significant improvement can be detected in correspondence of *Ntot* = 100 with respect to *Ntot* = 100. Moreover, starting from 100 filters, i.e., given that enough filters were used, as this number increases, the positive influence of adding filters disappears. Even if for English the result for 300 filters was slightly better than that with 100, for both languages similar trends can be recognized, therefore this difference can be ascribed to the experimental variance. On the contrary, a decreasing trend of the accuracy can be detected as *Ntot* increases.

These results mean that a minimum of 100 filters should be used, since at least 100 filters were useful to extract different features from text. Moreover, the decreasing trend can be explained by observing that each filter adds 609 weights to the model, therefore adding a great number of filters cause overfitting on training data, and thus a worse accuracy on testing. Therefore, the value *Ntot* = 100 appears the best choice.

As far as the activation function is regarded, those reported in Table 12, together with respective results, are analyzed.

Table 12. Testing accuracy obtained by different activation functions [1].

f		eLU	Identity	ReLU	Sigmoid	Softplus	Softsign	Tanh
Acc (%)	**English**	84.9	84.7	85.1	65.6	85.0	79.0	83.4
	Italian	84.1	84.0	83.7	56.7	84.3	79.2	82.9

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, *batch* = 100, P_{keep} = 1, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

These results, similar for both languages, show that a very low accuracy was obtained by using the sigmoid function. Also with f = softsign the accuracy was significantly lower than the others, while using f = tanh, the accuracy was better, but however, a t-test still reveals that the difference with the others (eLU, Identity, ReLU, softplus) was significant. On the other hand, the activation functions f = eLU, f = Identity, f = ReLU, and f = softplus allow to obtain higher accuracies, with variations among them comparable with the experimental variance. From Figure 2, it can be noticed that these functions giving better results can be distinguished by their characteristic of infinitely increasing trend, with respect to the worse ones that have asymptotic behavior. Since they offer comparable results, one of them can be chosen. For example, f = softplus could be chosen by considering resulting small differences in accuracy, while f = Identity could be preferred in order to design the simplest network architecture.

3.4. Learning Procedure

The first hyperparameter defining the learning procedure considered here was the batch size, i.e., the number of examples considered together to calculate the loss value. Given the training dataset, the variability range was *batch* ∈ [1,900], and all the orders of magnitude were considered, i.e., *batch* = {1,10,100,900}. Results are reported in Table 13 and graphically represented in logarithmic scale in Figure 8.

Table 13. Testing accuracy obtained by different batch sizes [1].

batch		1	10	100	900
Acc (%)	**English**	88.1	88.1	85.1	80.2
	Italian	86.4	86.6	83.3	80.4

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, f = ReLU, P_{keep} = 0.5, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

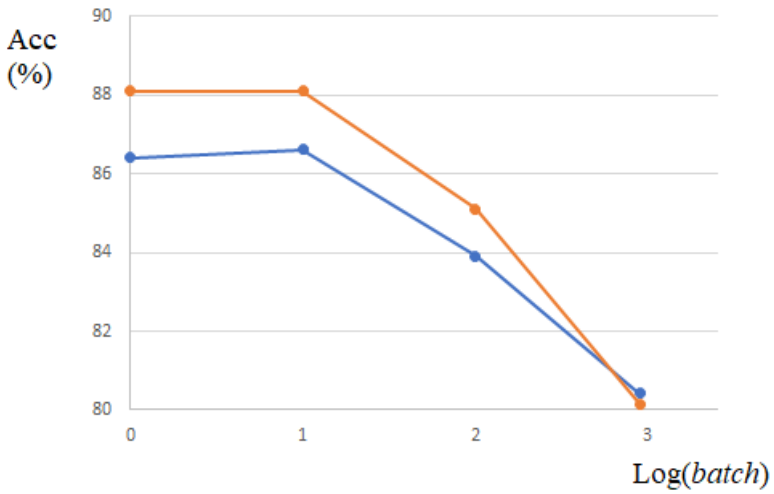

Figure 8. Testing accuracy obtained by different batch sizes, for English (**red**) and Italian (**blue**).

Figure 8 clearly shows that, while the influence of batch size was not significant for sizes between 1 and 10, as the batch size increases, the accuracy clearly decreases, with a very strong effect of this hyperparameter. This finding confirms the usefulness of employing batches instead of summing up the loss function for all the examples. In particular, the smaller the batch size was, the higher accuracy was obtained. However, one should also take into account that smaller batch sizes also cause much longer training time. For this reason, here, in order to choose the best batch size, between 1 and 10, having comparable performances, *batch* = 10 was chosen.

The dropout was also varied, in the range P_{keep} = (0,1], and in particular, in correspondence of the following representative levels: P_{keep} = {0.1,0.5,0.9,1}. Results are reported in Table 14.

Table 14. Testing accuracy obtained by different dropout [1].

P_{keep}		0.1	0.5	0.9	1.0
Acc (%)	**English**	82.6	84.8	84.9	84.8
	Italian	81.9	83.5	84.2	84.2

[1] Other settings were: *AvoidPunctuation* = False, *WEinit* = pre-trained, *De* = 300,*WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, *f* = ReLU, *batch* = 100, *l2* = 3.0, *optimizer* = Adadelta, η = 0.1.

From Table 14, it can be seen that, excepting the case P_{keep} = 0.1, which causes significant accuracy worsening, the other cases were very similar. This means that for this kind of system, and for the considered size of the dataset, dropout was not strictly necessary. Therefore, for the considered dataset, the dropout can be avoided, by choosing P_{keep} = 1.0, or equivalently P_{keep} = 0.9 can be chosen.

The regularization term *l2* was also considered, at the following levels: *l2* = {1.0,3.0,5.0}. Results are reported in Table 15.

Table 15. Testing accuracy obtained by different regularization terms [1].

l2		1.0	3.0	5.0
Acc (%)	**English**	86.7	85.1	83.4
	Italian	85.6	83.9	83.5

[1] Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, *f* = ReLU, *batch* = 100, P_{keep} = 0.5, *optimizer* = Adadelta, η = 0.1.

From Table 15, a slight but significant decreasing trend of the accuracy can be detected while *l2* increases. Therefore, *l2* = 1.0 was chosen.

Finally, the updating rule *optimizer* used to perform weights update by SGD backpropagation algorithm, and the associated learning rate η, were studied together, in order to evaluate also their probable interactions. According to a full factorial design, all the combinations of factors levels were experimented, i.e., all the available updating rules *optimizer* = {*Adadelta,Adagrad,Adam,Ftrl, GradientDescent,Momentum,ProximalAdagrad,ProximalGradientDescent,RMSProp*} combined with all the magnitude orders in the considered range of the learning rate η= {0.01,0.1,1,10}. Results were reported in Table 16 and shown in Figure 9. In Table 16, some results were not reported ("-"), since the corresponding experiments were not performed, because they make no sense in light of the other experiments. Moreover, the results of some experiments were reported as "<20.0", since in those cases the learning procedure did not offer acceptable accuracy. Some others were reported as "DIV", since the learning procedure gave exceptions due to overflow. The results indicated by an asterisk were obtained by 20,000 epochs instead of 2000; however, results with different numbers of epochs can be compared, because in all the cases the training was stopped after that convergence was reached. For each set of experiments with different learning rates, the best result is reported in bold.

From Table 16, it can be evinced that some updating rules does not work with a too high learning rate, giving overflow problems. In particular, when *optimizer* = {GradientDescent,Momentum, ProximalGradientDescent}, the learning rate η= 1 was already too high. Moreover, when *optimizer* = Ftrl and η= 0.01 (too low η), and when *optimizer* = RMSProp and η= 1 (too high ηin this case), the learning procedure does not improve the testing accuracy of the initial random model. Another point to take into account was that experiments obtained by 20000 epochs, necessary to get convergence for some low values of η, need much more computation time. These observations limit the range of the usable values of η, peculiarly for each updating rule.

Table 16. Testing accuracy obtained by different learning updating rules and learning rates [1].

		optimizer\η	0.01	0.1	1	10
Acc (%)	English	**Adadelta**	84.9 *	87.2 *	**88.4**	88.2
		Adagrad	85.8 *	**86.1**	55.7	-
		Adam	**85.1**	65.8	50.5	-
		Ftrl	<20.0	**57.4**	53.5	-
		GradientDescent	**87.4** *	85.7	DIV	-
		Momentum	**87.2** *	86.4	DIV	-
		ProximalAdagrad	85.4 *	**86.6**	57.1	-
		ProximalGradientDescent	**87.0** *	85.3	DIV	-
		RMSProp	**87.0**	55.3	<20.0	-
	Italian	**Adadelta**	83.8 *	87.4 *	**88.6**	88.0
		Adagrad	85.2 *	**86.0**	57.0	-
		Adam	**85.9**	67.8	66.4	-
		Ftrl	<20.0	**56.4**	49.8	-
		GradientDescent	**86.2** *	82.3	DIV	-
		Momentum	**86.2** *	82.8	DIV	-
		ProximalAdagrad	85.0 *	**85.3**	63.6	-
		ProximalGradientDescent	**86.0** *	82.5	DIV	-
		RMSProp	**86.4**	57.0	<20.0	-

[1] Experiments not performed reported as "-". Not acceptable accuracy reported as "<20.0". Exceptions due to overflow reported as "DIV". Results obtained by 20,000 epochs instead of 2000 indicated by an asterisk. Best result for each set of experiments with different learning rates reported in bold. Other settings were: *AvoidPunctuation* = True, *WEinit* = pre-trained, *De* = 300, *WEtuning* = dynamic, *Fs* = {1,2,3}, *Ntot* = 300, *f* = softplus, *batch* = 100, P_{keep} = 1.0, *l2* = 3.0.

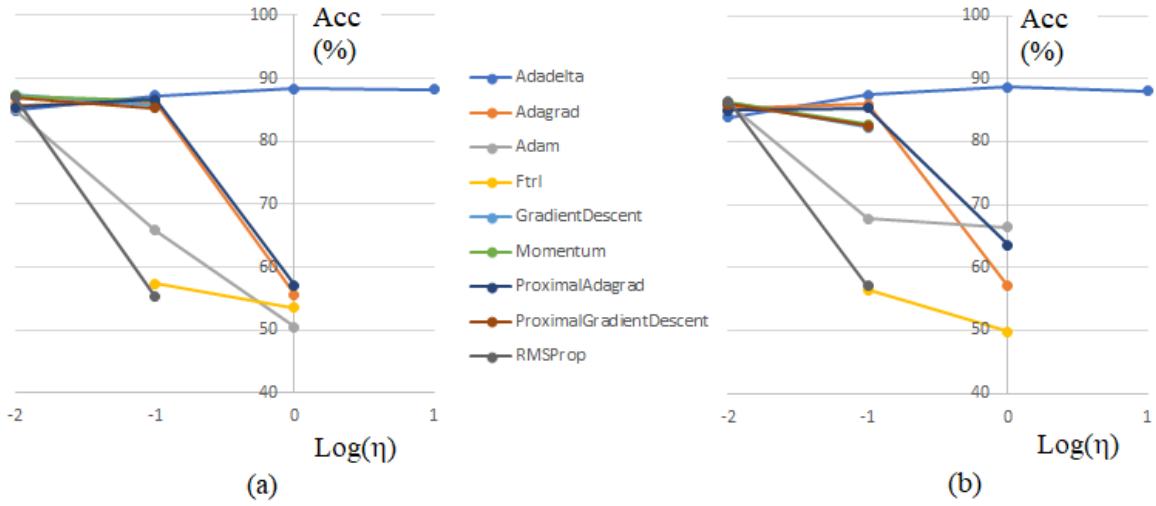

Figure 9. Testing accuracy obtained by different weights updating rules and different learning rates, for English (**a**) and Italian (**b**).

Most updating rules result equivalent for low learning rates, as can be evinced by Figure 9, in correspondence of η = 0.01. In particular, the cases *optimizer* = {GradientDescent, Momentum,ProximalGradientDescent} result equivalent for this dataset in the whole range of η. Moreover, as can be seen in Figure 9, in the acceptable ranges of η for each updating rule, most of them present similar trends, with significantly increasing accuracy values as η decreases. This can be explained by the network behavior of adapting fast to training data for high learning rates, which allows to increase predictivity only during the first few epochs. Therefore, while experimenting lower learning rates was not doable due to too high computation time, the option of higher learning rates was not promising for most of the cases. On the other hand, when *optimizer* = Adadelta, accuracy surprisingly increases with η, even if the differences in the range η = [0.1,10] were comparable with experimental variance, and this allows using high learning rates (e.g., η = 1), and not too many epochs. Moreover, in

correspondence of *optimizer* = Adadelta and $\eta \geq 0.1$, a significant accuracy improvement was gained, with respect to the maximal values of the other algorithms obtained with $\eta = 0.01$. Therefore, the following couple of values of the considered degrees of freedom was chosen as optimal: *optimizer* = Adadelta and $\eta = 1$.

3.5. Most Influencing Hyperparameters

The previous findings can be compared and summarized as follows.

For classifying questions (in 9 classes, using 900 training instances, with a CNN), the influence associated to variations of different settings, relative to text representation, CNN architecture, and learning procedure, was qualitatively described in Table 17.

Table 17. Qualitative description of influence of settings.

Setting	Symbol	Influence
Words representation		
Eliminate punctuation	*AvoidPunctuation*	Not significant
Use of pre-trained and fine-tuned Word Embedding vectors	*WEinit* AND *WEtuning*	Significant
Embedding dimension	*De*	Very strong ($De \leq 100$) – Significant ($100 < De \leq 300$) – Not significant ($300 < De \leq 500$)
CNN architecture		
Filter size	*Fs*	Significant
Total number of filters	*Ntot*	Significant ($Ntot \leq 100$) – Not significant ($Ntot > 100$)
Activation function	*f*	Not significant (among eLU,Identity,ReLU,softPlus) – Significant (vs. softsign,tanh) – Very strong (vs. sigmoid)
Learning procedure		
Batch size	*batch*	Not significant ($batch \leq 10$) – Strong ($batch > 10$)
Dropout	P_{keep}	Significant ($P_{keep} < 0.5$) – Not significant ($P_{keep} \geq 0.5$)
Loss regularization	*l2*	Significant
Weights updating rule	*optimizer*	Not significant (associated with $\eta = 0.01$) – Very strong (associated with $\eta \geq 0.1$)
Learning rate	η	Significant (associated with *optimizer* = Adadelta) – Very strong (otherwise)

From Table 17, the set of possible causes of very bad results can be individuated, i.e., too few embedding dimensions, sigmoid activation function, and a wrong choice of learning rate associated with a certain weights updating rule.

On the other hand, the strongest positive effect on accuracy was associated with a small batch size. Other settings give significant positive effects: Use of pre-trained and fine-tuned WE vectors, minimum 300 embedding dimensions, filter size equal to 2, minimum 100 total number of filters, choice of the activation function among {eLU,Identity,ReLU,softplus}, low loss regularization constant, and low learning rate. On the contrary, the influence of eliminating punctuation, and of the dropout function (given $P_{keep} \geq 0.5$) were not significant.

3.6. Experiments with Optimal Settings

The results reported above allow individuating the best settings, for hopefully obtaining the highest accuracy values, with respect to those reported so far.

Therefore, some experiments were performed in correspondence of the best settings, in CV and with some repetitions, in order to validate the optimization procedure described before. In particular, two different settings were chosen, one (OPT1) comprising one of the best activation functions (f = softplus), the other without the activation function (f = Identity).

Moreover, the results obtained here were compared with those obtained with settings individuated as optimal in previous works [19,21]. For configurations found in previous works, fastText WE pre-trained vectors are used here.

In Table 18, the settings relative to different final experiments are reported, together with respective accuracy on testing.

Table 18. Testing accuracy for optimal settings, averaged on experiments repetitions, and for settings individuated in previous literature.

Symbol	OPT1 (This Work)	OPT2 (This Work)	[21]	[19]
Ntrain/Ntest	900/100			
Words representation				
AvoidPunctuation	True	True	False	False
WEinit	pre-trained	pre-trained	pre-trained	pre-trained
De	300	300	300	300
WEtuning	dynamic	dynamic	dynamic	dynamic
CNN architecture				
Fs	2	2	{3,4,5}	{2,3,4,5}
Ntot	100	100	300	400
f	softplus	Identity	ReLU	ReLU
Learning procedure				
batch	10	10	50	50
P_{keep}	1	1	0.5	0.7
l2	1.0	1.0	3.0	5.0
optimizer	Adadelta	Adadelta	Adadelta	Adadelta
η	1.0	1.0	0.1	0.1
Performance				
Acc (%) for English	88.8	89.2	85.6	85.7
Acc (%) for Italian	89.0	89.0	85.4	85.0

Results presented in Table 18, firstly validate the optimization performed of the whole model. Indeed, the accuracy values were the highest obtained so far on this dataset.

Moreover, the results obtained by taking into account optimal configurations individuated by [19,21] were significantly worse than those obtained here. The most noticeable differences in the configurations revealed that the model chosen here was much simpler, since it avoids considering punctuation, uses only 100 filters of size 2, and does not use dropout (nor any activation function, for OPT2).

Since [19,21] found their best configurations on a different dataset, the same comparison was performed on the most used dataset provided by [35] for the English language. In Table 19, the settings and the respective accuracy on testing were reported, relative to different final experiments, also on this bigger sized dataset.

Table 19. Testing accuracy on [35] data with optimal settings, and with settings individuated in previous literature.

Symbol	OPT1 (This Work)	OPT2 (This Work)	[21]	[19]
Ntrain/Ntest	5452/500			
Words representation				
AvoidPunctuation	True	True	False	False
WEinit	pre-trained	pre-trained	pre-trained	pre-trained
De	300	300	300	300
WEtuning	dynamic	dynamic	dynamic	dynamic
CNN architecture				
Fs	2	2	{3,4,5}	{2,3,4,5}
Ntot	100	100	300	400
f	softplus	Identity	ReLU	ReLU
Learning procedure				
batch	10	10	50	50
P_{keep}	1	1	0.5	0.7
l2	1.0	1.0	3.0	5.0
optimizer	Adadelta	Adadelta	Adadelta	Adadelta
η	1.0	1.0	0.1	0.1
Performance				
Acc (%) for English	93.0	92.2	91.8	91.0

Among results of Table 19, those obtained with the proposed optimal settings were better than those obtained with settings optimized in previous works for this particular dataset. This confirms the validity and transferability of the optimal text representation, CNN architecture, and learning procedure obtained here for the QC task.

3.7. Limitations

The optimal settings found here were based on a multilingual dataset regarding QC, using the taxonomy explained in Section 2.1. Moreover, they were validated on a further dataset, also regarding QC, presenting a different taxonomy.

However, the optimality of those settings cannot be demonstrated for any taxonomy of question classes. Moreover, it cannot be extended to other sentence classification tasks. For example, if a filter of size 2 was enough to classify some questions by just individuating "How much" sequence of words, the same small filter could be undersized to distinguish more fine-grained question classes or to classify sentiment of affirmative sentences.

4. Conclusions

This paper presented a study performed to analyze the settings of Convolutional Neural Networks for Question Classification, in terms of words representation, network architecture and learning procedure.

Both English and Italian languages were considered, since they have different morphological richness, and training sets made of different number of questions were tested. All experiments were based on questions properly extracted from the same multilingual dataset, in order to check possible dependencies of optimal settings with respect to language.

All the hyperparameters and the most plausible interactions among them were tested in correspondence of wide ranges of variability. For each of them, statistical significance of its influence was evaluated by means of a comparison with intrinsic variability, measured through repetitions of the same experiments.

Results of the huge number of experiments drove to the individuation of optimal settings, which are similar for both languages. They can be summarized as follows. Regarding the text representation, it is better to avoid punctuation, to use pre-trained word embedding vectors with dimension 300, and fine-tune them according to available data; regarding the architecture, 100 filters of size 2 were enough for coarse-grain classification, and an infinitely increasing activation function should be preferred (eLU, ReLU, softplus), or equivalently no activation function (Identity); regarding the learning procedure, using a small batch of 10 gives strong improvements, while choosing it smaller only increases computation time, dropout and loss regularization should be avoided, and the best and fastest optimizer was Adadelta, associated with learning rate 1.0.

The individuated best configuration was tested on the same data and on a different set of questions widely used for QC, and compared to the configurations suggested by the most relevant previous works. These further results validated the optimization performed and confirmed the transferability of the best settings on different data, since in all cases the models optimized here showed significantly better classification accuracy than those suggested before.

Author Contributions: Conceptualization, M.P., M.E. and H.F.; methodology, M.P. and M.E.; software, M.P.; validation, M.E. and H.F.; formal analysis, M.P.; investigation, M.P. and M.E.; resources, G.D.P.; data curation, M.P.; writing—original draft preparation, M.P. and M.E.; writing—review and editing, M.P. and M.E.; visualization, M.P.; supervision, M.E., G.D.P. and H.F.; project administration, G.D.P.; funding acquisition, G.D.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yadav, A.; Vishwakarma, D.K. Sentiment analysis using deep learning architectures: A review. *Artif. Intell. Rev.* **2019**. [CrossRef]
2. Yuan, S.; Zhang, Y.; Tang, J.; Hall, W.; Cabotà, J.B. Expert finding in community question answering: A review. *Artif. Intell. Rev.* **2020**, *53*, 843–874. [CrossRef]
3. Wang, Y.; Wang, M.; Fujita, H. Word Sense Disambiguation: A comprehensive knowledge exploitation framework. *Knowl. Based Syst.* **2020**, *190*, 105030. [CrossRef]
4. Pota, M.; Fuggi, A.; Esposito, M.; De Pietro, G. Extracting Compact Sets of Features for Question Classification in Cognitive Systems: A Comparative Study. In Proceedings of the 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3rd Workshop on Cloud and Distributed System Applications, Krakow, Poland, 4–6 November 2015; IEEE: Piscataway, NJ, USA; pp. 551–556. [CrossRef]
5. Pota, M.; Esposito, M.; De Pietro, G. A forward-selection algorithm for SVM-based question classification in cognitive systems. In Proceedings of the 9th International KES Conference on Intelligent Interactive Multimedia: Systems and Services (KES-IIMSS-16), Tenerife, Spain, 15–17 June 2016; pp. 587–598. [CrossRef]
6. Pota, M.; Esposito, M.; De Pietro, G. Learning to rank answers to closed-domain questions by using fuzzy logic. In Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, Italy, 9–12 July 2017; pp. 1–6. [CrossRef]
7. Argamon, S.; Koppel, M.; Pennebaker, J.W.; Schler, J. Automatically profiling the author of an anonymous text. *Commun. ACM* **2009**, *52*, 119–123. [CrossRef]
8. Estival, D.; Gaustad, T.; Pham, S.B.; Radford, W.; Hutchinson, B. Tat: An author profiling tool with application to arabic emails. In Proceedings of the Australasian Language Technology Workshop, Melbourne, Australia, 21–30 December 2007.
9. Franco-Salvador, M.; Rangel, F.; Rosso, P.; Taulé, M.; Martí, M.A. Language variety identification using distributed representations of words and documents. In Proceedings of the CLEF 2015 Conference and Labs of the Evaluation Forum-Experimental IR meets Multilinguality, Multimodality, and Interaction, LNCS, Toulouse, France, 8–11 September 2015; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9283, pp. 24–40.
10. Bayot, R.; Gonçalves, T. Author Profiling using SVMs and Word Embedding Averages—Notebook for PAN at CLEF 2016. In Proceedings of the Working Notes of CLEF'2016—Conference and Labs of the Evaluation forum CLEF 2016 Evaluation Labs and Workshop—Working Notes Papers, Évora, Portugal, 5–8 September 2016.

11. Liu, G.; Guo, J. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* **2019**, *337*, 325–338. [CrossRef]
12. Guo, B.; Zhang, C.; Liu, J.; Ma, X. Improving text classification with weighted word embeddings via a multi-channel TextCNN model. *Neurocomputing* **2019**, *363*, 366–374. [CrossRef]
13. Wang, P.; Xu, B.; Xu, J.; Tian, G.; Liu, C.-L.; Hao, H. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing* **2016**, *174*, 806–814. [CrossRef]
14. Poria, S.; Peng, H.; Hussain, A.; Howard, N.; Cambria, E. Ensemble application of convolutional neural networks and multiple kernel learning for multimodal sentiment analysis. *Neurocomputing* **2017**, *261*, 217–230. [CrossRef]
15. Xia, W.; Zhu, W.; Liao, B.; Chen, M.; Cai, L.; Huang, L. Novel architecture for long short-term memory used in question classification. *Neurocomputing* **2018**, *299*, 20–31. [CrossRef]
16. Loni, B. *A Survey of State-of-the-Art Methods on Question Classification*; Technical Report; Delft University of Technology: Delft, The Netherlands, 2011.
17. Dale, R. Classical approaches to natural language processing. In *Handbook of Natural Language Processing*; Chapman & Hall/CRC: Boca Raton, FL, USA, 2010.
18. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
19. Zhang, Y.; Wallace, B.C. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In Proceedings of the 8th International Joint Conference on Natural Language Processing, Taipei, Taiwan, 26–31 July 2015; pp. 253–263.
20. Pota, M.; Esposito, M.; De Pietro, G. Convolutional Neural Networks for Question Classification in Italian Language. In Proceedings of the 16th International Conference on Intelligent Software Methodologies, Tools, and Techniques (SOMET_17), Kitakyushu, Japan, 26–28 September 2017; pp. 604–615. [CrossRef]
21. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
22. Qin, P.; Xu, W.; Guo, J. An empirical convolutional neural network approach for semantic relation classification. *Neurocomputing* **2016**, *190*, 1–9. [CrossRef]
23. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, New York, NY, USA, 5–9 July 2008; pp. 160–167.
24. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
25. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014.
26. Yin, W.; Schütze, H. Multichannel variable-size convolution for sentence classification. In Proceedings of the 19th Conference on Computational Language Learning, Beijing, China, 30–31 July 2015; pp. 204–214.
27. Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. A neural probabilistic language model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
28. Dos Santos, C.N.; Gatti, M. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland, 23–29 August 2014.
29. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014.
30. Conneau, A.; Schwenk, H.; Barrault, L.; LeCun, Y. Very deep convolutional networks for natural language processing. *arXiv* **2016**, arXiv:1606.01781.
31. Dachapally, P.R.; Ramanam, S. In-depth Question classification using Convolutional Neural Networks. *arXiv* **2018**, arXiv:1804.00968.
32. Lei, T.; Shi, Z.; Liu, D.; Yang, L.; Zhu, F. A novel CNN-based method for Question Classification in Intelligent Question Answering. In Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence, Sanya, China, 21–23 December 2018.

33. Pota, M.; Esposito, M. Question Classification by Convolutional Neural Networks Embodying Subword Information. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018. [CrossRef]
34. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Chen, T. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
35. Li, X.; Roth, D. Learning question classifiers. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), Morristown, NJ, USA, 26–30 August 2002.
36. Robbins, H.; Monro, S. A Stochastic Approximation Method. *Ann. Math. Stat.* **1951**, *22*, 400. [CrossRef]
37. Bottou, L.; Bousquet, O. The Tradeoffs of Large Scale Learning. In *Optimization for Machine Learning*; MIT Press: Cambridge, MA, USA, 2012; pp. 351–368.
38. Bottou, L. Online Algorithms and Stochastic Approximations. In *Online Learning and Neural Networks*; Cambridge University Press: Cambridge, UK, 1998.
39. Neyman, J. Outline of a theory of statistical estimation based on the classical theory of probability. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Sci.* **1937**, *236*, 333–380.