



A Deep Learning Framework for Italian Text Classification Using Generalized Language Models

Savita ^{1*}, Dilip Kumar ²

¹⁻² Department of Computer Science and Engineering, Vishveshwarya Group of Institutions, Greater Noida, India

* Corresponding Author: Savita

Article Info

ISSN (Online): 3107-6580

Volume: 02

Issue: 05

Received: 10-07-2026

Accepted: 08-08-2026

Published: 06-09-2026

Page No: 21-25

Abstract

This study investigates deep-learning approaches to the problem of text classification, with a specific focus on an Italian-language corpus. Text classification is a central task in natural language processing in which predefined categories are assigned to units of text. Because of the discrete nature of language, effective pre-processing and the transformation of text into numerical representations are essential for enabling learning models to capture meaningful patterns. We examine multiple strategies for word representation and emphasise the role of word embeddings in model performance. In particular, two widely used pre-trained embedding methods—Word2Vec and ELMo—are compared, together with an alternative strategy in which embeddings are learned dynamically during training. For the classification task, both Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) recurrent networks are employed as feature extractors. Five models are trained and evaluated on a real-world multi-label dataset derived from Italian collective labour agreements. The results provide a comparative analysis of embedding techniques and neural architectures, highlighting their strengths and limitations and, in particular, the trade-off between predictive quality and computational cost.

DOI: <https://doi.org/10.54660/IJECA.2026.2.5.21-25>

Keywords: text classification, multi-label classification, deep learning, word embeddings, ELMo, Word2Vec, CNN, LSTM, Italian NLP

1. Introduction

Text classification—the assignment of one or more predefined categories to a document—is one of the most common and practically important tasks in natural language processing (NLP). It underpins applications such as topic labelling, content filtering and the automatic organisation of large document collections. The task is challenging because natural language is discrete, high-dimensional and context-dependent, so a document must first be converted into a numerical form that a learning algorithm can process while retaining its semantic content.

This work addresses a real-world, multi-label variant of the problem using a corpus of Italian collective national labour agreements (CCNL). Each text may legitimately belong to several of fourteen thematic categories at once, which makes the task substantially harder than ordinary single-label classification. The central question is whether modern generalized language models—specifically contextual embeddings produced by ELMo—offer a worthwhile advantage over simpler representations such as task-trained embeddings or pre-trained Word2Vec vectors, once the additional computational cost of the richer models is taken into account.

To answer this, five deep-learning classifiers are constructed by combining three embedding strategies (task-trained, Word2Vec and ELMo) with two feature extractors (a multi-filter CNN and a bidirectional LSTM). All models are trained under identical conditions and compared on precision, recall, F1-score and an exact-match “hard accuracy” metric, as well as on training time.

The remainder of the paper reviews the relevant background (Section 2), describes the dataset (Section 3) and methodology (Section 4), reports the experimental results (Section 5), discusses their implications (Section 6) and concludes (Section 7).

2. Background and Related Work

Classical treatments of text categorisation frame the problem as supervised learning over sparse bag-of-words representations [1]. Deep-learning methods instead learn dense, distributed representations of words and compose them into document-level features. Two families of architectures are especially relevant here. Convolutional Neural Networks apply filters of several widths across a sentence to capture salient local n -gram patterns, while Recurrent Neural Networks—particularly the LSTM variant—model sequential dependencies and are well suited to capturing longer-range context [2].

Equally important is the choice of word representation. Word2Vec learns a fixed vector for every word in a vocabulary from co-occurrence statistics, producing embeddings that are task-independent and reusable but insensitive to context: a word has the same vector regardless of the sentence it appears in. Generalized (contextual) language models such as ELMo address this limitation by producing representations that depend on the entire surrounding sentence through a bidirectional language model, and by deriving the initial layer from character-level convolutions, which reduces dependence on a fixed

vocabulary and allows meaningful embeddings for unseen words [4]. These properties make contextual embeddings attractive for transfer learning, at the price of greater model depth and training cost—the trade-off this study sets out to quantify.

3. Dataset and Exploratory Analysis

3.1. Task Definition

The corpus is a portion of the Italian CCNL agreements, segmented into paragraph-level texts. Each instance is a pair (x_i, y_i) , where x_i is the processed text and y_i is a fourteen-dimensional binary vector whose j -th component equals 1 when the text belongs to category j . The objective is to predict, as accurately as possible, the conditional probability that a text belongs to each of the 14 classes. Because a single text may carry several labels, the problem is one of multi-label classification.

3.2. Word-Length and Label Distribution

A descriptive analysis guided several modelling decisions. Figure 1 shows the distribution of word counts: most paragraphs contain fewer than 200 words, and only about 25% exceed 200 words. A small number of very long texts suggests that some contracts were incorrectly segmented. To keep memory requirements tractable, a maximum length threshold of $t = 300$ words was imposed and longer texts were discarded; preliminary experiments showed that raising or removing this threshold forced impractical RAM usage even at very small batch sizes.

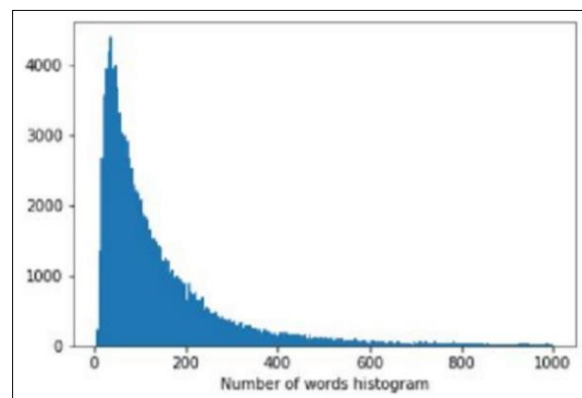


Fig 1: Distribution of the number of words per text in the dataset.

Figure 2 shows that the label distribution is strongly imbalanced. The TE and CS categories dominate the corpus, whereas OD and GE are represented by very few examples.

This imbalance motivates the use of class weights in the loss function (Section 4.3) and, as the results confirm, explains the weaker performance observed on the rarest classes.

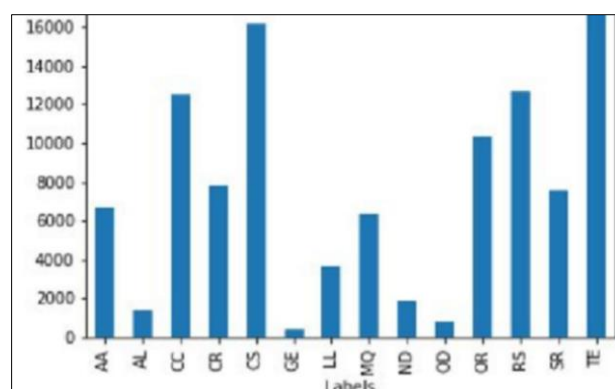


Fig 2: Distribution of the 14 labels across the dataset.

3.3. Train / Validation / Test Protocol

The data were split while respecting both class imbalance and the multi-label structure. For every class, 80% of its instances were assigned to training and 20% to testing; the training portion was then split again to produce a refined training set and a validation set. Crucially, multi-label instances

appearing in training were excluded from the validation and test sets to prevent overlap that would otherwise yield over-optimistic estimates. As a side effect, multi-label examples are slightly under-represented in validation and test, but such cases are rare in any event. Table 1 summarises the resulting composition.

Table 1: Dataset composition by number of labels per text.

Label cardinality	Training	Validation	Test	Whole dataset
One label	46,416	11,611	14,489	72,516
Two labels	12,409	1,262	619	14,290
Three or more labels	1,186	43	6	1,235
Total examples	60,011	12,916	15,114	88,041

4. Methodology

4.1. Pre-Processing

The pre-processing pipeline comprises tokenisation, lower-casing and the removal of stop words. A regular-expression tokeniser separates words, numbers, punctuation and special characters. Words are retained as-is; dates, abbreviations and generic numbers are replaced by the placeholder tokens `xxxdatexxx`, `xxxdlxxx` and `xxxnumxxx` respectively, preserving numeric information without inflating the vocabulary. All tokens are then lower-cased, which reduces vocabulary size at the cost of some ambiguity—an issue for fixed-vocabulary models such as Word2Vec but not for contextual models such as ELMo. Finally, Italian stop words are removed using the standard NLTK list^[3].

4.2. Model Architectures

Every model follows a common template. A series of tokens

is first mapped by an embedding function E into a matrix of word vectors; a feature-extraction stage F then produces a document vector f , which a final single-layer perceptron projects onto a 14-dimensional output. A sigmoid activation converts each output into an independent probability, as required for multi-label prediction. Combining three embedding strategies with two feature extractors yields the five models summarised in Table 2. The task-trained embeddings use a dimension of 300 to match the Word2Vec vectors; the multi-filter CNN output is mapped through an additional single-layer MLP to a 512-dimensional space so that it matches the output size of the bidirectional LSTM. The Word2Vec vectors are taken from a Wikipedia2Vec model for Italian, and the ELMo model is the Italian version of Schuster *et al.*^[4], whose embedding network produces 1024-dimensional contextual vectors and is fine-tuned with dropout during training.

Table 2: Summary of the five classification models.

Model	Embedding	Feature extractor	Embed. dim	Notes
TrainableCNN	Task-trained	Multi-filter CNN	300	Filters 2,3,4,5; 256 filters; proj. 512
TrainableLSTM	Task-trained	Bidirectional LSTM	300	Feature dim 256
Word2VecCNN	Word2Vec (frozen)	Multi-filter CNN	300	Wikipedia2Vec, Italian
Word2VecLSTM	Word2Vec (frozen)	Bidirectional LSTM	300	Wikipedia2Vec, Italian
ELMoLSTM	ELMo (fine-tuned)	Bidirectional LSTM	1024	Contextual; dropout applied

4.3. Training and Prediction

All five models were trained under identical settings. Binary cross-entropy (BCE), averaged over classes, served as the loss function. To counter class imbalance, each class was assigned a weight derived from the inverse ratio of its sample count to that of the most populous class, refined by a local grid search; the rarest classes (GE, OD, AL) therefore received the largest weights. Optimisation used Adam with a batch size of 32. An early-stopping strategy retained the model with the lowest validation loss, with a maximum of 20 epochs and a patience of 4. Training was performed on an Azure virtual machine equipped with an NVIDIA Tesla K80 GPU, using the AllenNLP library on top of PyTorch. At prediction time the multi-label task is decomposed into fourteen binary decisions, a class being assigned when its

predicted probability exceeds a per-class threshold tuned on the validation set.

5. Results

Training cost varied widely across models. The single-embedding CNN models trained in minutes, LSTM-based models took longer, and the ELMoLSTM model required roughly 11.5 hours even with GPU acceleration. Notably, although each ELMoLSTM epoch is far more expensive, the model converged in fewer epochs than the other LSTM models. Figure 3 shows a representative training curve: training and validation loss both fall to roughly 0.1, and the red line marks the epoch of minimum validation loss selected by early stopping.

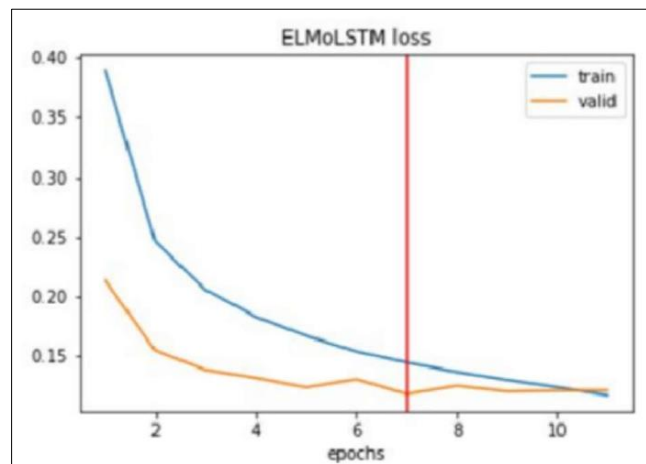


Fig 3: Training and validation loss for the ELMoLSTM model; the vertical line marks the early-stopping epoch.

Table 3 reports, for each model, the training cost together with the weighted-average precision, recall and F1 on the validation set. On the validation set ELMoLSTM attains the

highest precision, but only marginally, and no single model dominates recall. The models are close in overall F1, with TrainableLSTM slightly ahead.

Table 3: Training cost and weighted-average validation metrics.

Model	Cutoff / total epochs	Training time	Precision	Recall	F1
Word2VecCNN	5 / 9	7 min	0.864	0.891	0.877
TrainableCNN	4 / 8	13 min	0.861	0.895	0.877
TrainableLSTM	9 / 13	31 min	0.872	0.892	0.882
Word2VecLSTM	15 / 19	42 min	0.871	0.883	0.876
ELMoLSTM	7 / 11	11 h 25 min	0.873	0.877	0.875

Because F1 balances precision and recall, it provides the most informative single comparison. The exact-match hard-accuracy metric is more demanding: a text counts as correct only when every one of its true labels is predicted and no extra label is assigned. Hard accuracy on the validation set averaged around 0.80 for all models, but fell sharply as the number of true labels per text increased—from about 0.85 for

single-label texts to below 0.10 for texts with three or more labels—reflecting how few such examples the models had seen. Table 4 reports the aggregate metrics on the held-out test set, which confirm the validation picture: the models are tightly clustered, with TrainableLSTM and Word2VecLSTM achieving the best test hard accuracy (0.839) and TrainableLSTM the best test F1 (0.893).

Table 4: Aggregate metrics on the test set.

Metric	TrainableCNN	TrainableLSTM	Word2VecCNN	Word2VecLSTM	ELMoLSTM
Hard accuracy	0.823	0.839	0.831	0.839	0.837
Precision	0.852	0.865	0.861	0.870	0.867
Recall	0.920	0.923	0.918	0.912	0.908
F1-score	0.884	0.893	0.888	0.890	0.887

6. Discussion

Two consistent patterns emerge. First, LSTM feature extractors slightly outperform CNNs at capturing document-level representations for this task. Second, and more importantly, the contextual ELMoLSTM model does not translate its richer representations into a meaningful accuracy advantage here: its results are essentially on par with the far cheaper TrainableLSTM and Word2Vec models. A likely explanation is that the corpus contains many semantically similar examples, so the task is largely within reach of simpler representations.

The practical consequence is a cost–benefit judgement. ELMoLSTM required roughly 12 hours of GPU training against minutes for the trainable-embedding CNN and well under an hour for the LSTM variants, for no appreciable gain in F1 or hard accuracy. In a business setting, where implementation time and compute cost must be weighed against quality, the simpler models are the rational choice for this dataset. This does not diminish the value of contextual

embeddings in general: ELMoLSTM converged in fewer epochs and would be expected to help in harder settings—scarce data, high lexical diversity, or more complex tasks such as question answering or machine translation. The weak results on the GE and OD classes further underline that data scarcity, rather than model capacity, is the dominant limiting factor here.

7. Conclusion

This paper presented and compared five deep-learning frameworks for multi-label classification of Italian collective labour agreements, spanning task-trained, Word2Vec and ELMo embeddings combined with CNN and LSTM feature extractors. On this corpus the model using a trainable embedding layer with an LSTM feature extractor proved the most effective overall choice once training cost is considered, and LSTM extractors showed a consistent edge over CNNs. The generalized language model ELMo performed on par with the simpler approaches while demanding far more

training time, making its additional complexity hard to justify for this particular task. The study illustrates both the strength of deep-learning methods for realistic, imbalanced, multi-label text classification and the importance of weighing predictive quality against computational cost when selecting a model for deployment.

References

1. Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys*. 2002;34(1):1-47.
2. Y. Goldberg, *Neural Network Methods for Natural Language Processing*. Morgan & Claypool, 2017.
3. S. Bird, E. Klein and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.
4. Schuster T, Ram O, Barzilay R, Globerson A. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *Proc. NAACL-HLT*. 2019.

How to Cite This Article

Savita, Kumar D. A deep learning framework for Italian text classification using generalized language models. *Int J Eng Comput Appl*. 2026;2(5):21-25.
doi:10.54660/IJECA.2026.2.5.21-25.

Creative Commons (CC) License

This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) License, which allows others to remix, tweak, and build upon the work non-commercially, as long as appropriate credit is given and the new creations are licensed under the identical terms.