Italian Lemmatisation with Neural Networks

Laurea Magistrale in Informatica

Report Natural Language Processing

A.A. 2022/2023

Filippo Bartolucci
filippo.bartolucci2@studio.unibo.it

Alfonso Esposito
alfonso.esposito5@studio.unibo.it

Contents

1. Introduction

The following project was completed as part of the exam for the **Natural Language Processing** course at the University of Bologna. The aim of the project was to tackle the challenge of lemmatisation for the Italian language. We followed the guidelines of the **EVALITA 2011** task and conducted experiments using the provided dataset. In this project, we will describe the machine learning algorithm we used, including a brief explanation of the neural network, and present the results of our experiments. Additionally, we will explore a novel model developed by OpenAI, derived from the GPT-3 family. Finally, we will offer a concise comparison between these two models.

1.1 Lemmatisation

Lemmatisation is a process in Natural Language Processing (NLP) where words are reduced to their base form, or lemma. The lemma is the root form of a word, which can be used to represent multiple inflected forms. This process is important in NLP as it helps in reducing the dimensionality of the text data and helps in standardizing the words to a common form.

$$\text{correndo} \rightarrow \text{correre}$$
$$\text{amici} \rightarrow \text{amico}$$

However, lemmatisation can be a challenging task, as it requires a deep understanding of the language grammar, context, and the relationship between words. One of the difficulties in lemmatisation is caused by ambiguous words that have multiple meanings or senses and can have different lemmas depending on the context in which they are used. Hence, lemmatisation is a crucial but challenging problem in NLP.

Lemmatisation and Parts of Speech (PoS) tagging are closely related in NLP as PoS tagging involves labeling words with their grammatical category. PoS information is often used to inform the lemmatisation process. As a result, lemmatisation is often considered a sub-product of PoS tagging.

1.2 Italian language

The process of lemmatization in Italian can prove to be challenging due to the language's intricate inflectional system and the potential for ambiguity in sentences. The richness of the inflectional system means that the same word can have multiple forms depending on context, making it difficult to identify the correct lemma. The possibility of ambiguity, where a single word can have different lemmas based on the context in which it appears, further complicates the task, making lemmatization in Italian a complex challenge to overcome

Word Form	**PoS-tag**	**Possible Lemmas**
cannone	NOUN	cannone, canna
morti	NOUN	morto, morte
regione	NOUN	regione, regia
aria	NOUN	aria, ario
macchina	NOUN	macchina, macchia
piccione	NOUN	piccione, piccia
matematica	NOUN	matematica, matematico
passano	VERB	passare, passire
danno	VERB	dannare, dare

Table 1.1: Example of ambigous words

2. Data and processing

EVALITA 2011 offers two datasets for solving the lemmatization task: the Development Set (DS) and the Test Set (TS)[1]. The DS consists of 17313 tokens from 703 sentences, while the TS contains 133756 tokens from 5596 sentences. Each token in both datasets is accompanied by its PoS tag and the corresponding lemma. The DS is utilized to train the neural network, and the TS is employed to evaluate the performance of the task.

2.1 Data processing

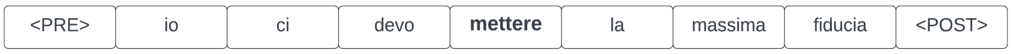
The lemma of a word is dependent on both its morphology and the context in which it is used. Thus, we chose to use a combination of the word form, the context surrounding it and the context PosTags as input to our neural network model. This approach takes into consideration both the morphological features and contextual information, ensuring a more accurate lemmatization result. By incorporating all of these factors, our model should be able to disambiguate words and provide a more accurate lemma prediction.

Word Form	**PoS-tag**	**Lemmas**	**Context**
albanese	ADJ	albanese	investimento spericolati faticavano a pagare gli interessi , il lek , la moneta **albanese**, aveva toccato i massimi sul dollaro .
agguati	NOUN	agguato	certo fatto con quell' invocazione : non &egrave; nel suo carattere temere gli **agguati**.
dolore	NOUN	dolore	loro soffrono il **dolore** del sentimento che dura sempre , notte e giorno ;

Table 2.1: Example of word context

2.2 Context encoding

To accurately determine the lemma of a word, it is necessary to understand its semantic meaning within a specific context. To account for this, our model considers the contextual information surrounding the target word. This context is defined as a fixed number of words on both sides of the target word, and is defined by special tokens indicating the beginning and end of the contextual window.



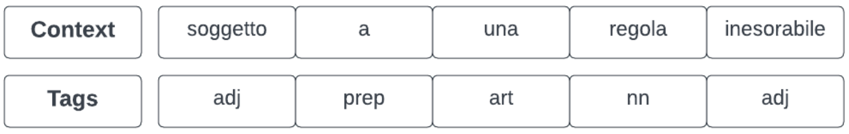
<PRE>	io	ci	devo	**mettere**	la	massima	fiducia	<POST>

Figure 2.1: Context

We encode each unique word as an integer for input into our neural network. This is accomplished by using word2vec to obtain the weights for our embedding layer, converting each word into a numerical representation that can be processed by the network.
Word2Vec is a model that learns dense representations of words in a high-dimensional space, so that words with similar meanings are close in the vector space. By initializing the weights of an embedding layer in our target model with word2vec, our model can take advantage of the pre-trained knowledge of word relationships learned by word2vec.

2.3 Tag encoding

In addition to the words, our model also takes into account a sequence of PoS tags for each word in the context window. To simplify the input for our model, we encoded each of the 31 unique PoS tags as a unique integer, treating each tag as a separate class in the model for processing and analysis.



Context	soggetto	a	una	regola	inesorabile
Tags	adj	prep	art	nn	adj

Figure 2.2: Context and tags sequence

2.4 Word encoding

In order to better understand the morphological structure of the target word, we chose to encode it character-by-character into integer values. This encoding method transforms the word into an array of integers, which can be processed and analyzed within our model. To further enhance the representation of the characters, we have incorporated an embedding layer within the model. This layer allows the model to learn the most important weights for each character, which results in more accurate predictions. By encoding the target word in this manner, we are able to take into account its morphological structure.

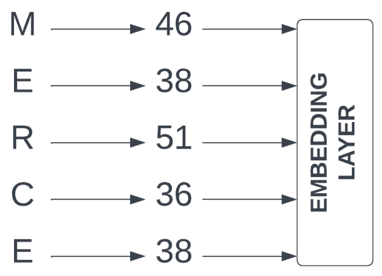


Figure 2.3: Example of word embedding at char level

2.5 Lemma encoding

In our quest to predict a lemma from context, word, and tag, we initially treated it as a multi-class classification problem. In this approach, each lemma was treated as a separate class. However, after conducting a series of experiments, it became clear that this method was not feasible. The main reason being that the number of classes was extremely high, which made it difficult to train the model with a limited number of samples. Furthermore, the test results showed that this method did not yield a desirable accuracy.

To overcome this challenge, we decided to change our approach and adopt a new strategy. We encoded each lemma character by character. This not only reduced the dimensionality of the output, but it also allowed us to treat the problem as a sequence-to-sequence task. The model was able to learn the relationships between the input and output in a more meaningful way. This change in approach significantly improved the results, allowing the model to better predict the lemma from the context, word, and tag.

3. Our Model

The proposed model architecture is a deep neural network that combines the context of a word, its corresponding tags, and the word itself as inputs. The model consists of three different branches, each of which processes a different input.

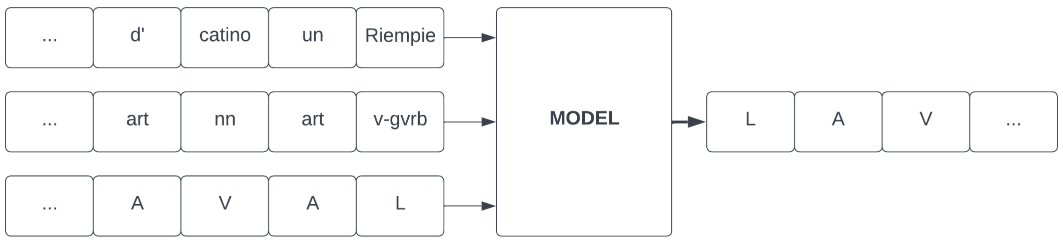


Figure 3.1: Inputs

3.1 Model Architecture

The first branch is for the context input which is passed through an embedding layer, which maps each word in the context to a high-dimensional vector representation. The embedding layer is initialized with pre-trained weights, and its training is set to be frozen. After the embedding, the context input is passed through a bidirectional LSTM layer, which encodes the context information into a compact representation.

The second branch is for the tag input. A second embedding layer is applied to the tag input, which maps each tag to a low-dimensional vector representation. The tag input is then passed through a bidirectional LSTM layer and the output is then passed through a TimeDistributed dense layer.

The third branch is for the word input. Then, the word input is passed through a third embedding layer, which maps each word to a high-dimensional vector representation. This embedding layer is trained during the model's training. The word input is then passed through a bidirectional LSTM layer.

The outputs of the three branches are then combined using an attention mechanism, which computes the weighted sum of the outputs of the context and tag branches. The attention mechanism helps the model to focus on the most important information. The result of the attention mechanism is concatenated with the word input, and the combined representation is passed through two consecutive bidirectional LSTM layers. This allows the model to capture complex relationships between the inputs.

Finally, the output of the second LSTM layer is passed through multiple dense layers, each with a dropout layer to prevent overfitting. The dropout layer randomly drops out a portion of the neurons during training, which helps to reduce overfitting. The final output of the model is a sequence of softmax activations, which represents the predicted lemma for each word in the input.

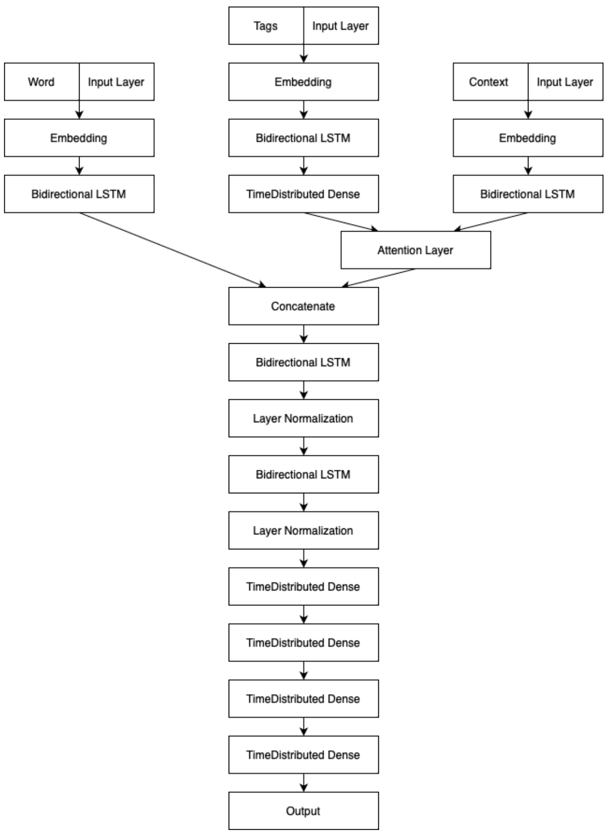


Figure 3.2: Our model

3.2 Custom Accuracy

To accurately evaluate the performance of our model in predicting lemmas character by character, we have created a custom accuracy function.

```
def accuracy(y_true, y_pred):
    y_true = tf.argmax(y_true, axis=-1)
    y_pred = tf.argmax(y_pred, axis=-1)
    correct_predictions = tf.reduce_all(tf.equal(y_true, y_pred), axis=-1)
    accuracy = tf.reduce_mean(tf.cast(correct_predictions, tf.float32))
    return accuracy
```

This function only considers a lemma prediction to be correct if every character in the generated sequence matches the expected result, rather than just evaluating based on the number of correctly predicted characters. The aim is to measure the accuracy of the lemma prediction as a whole, not just individual characters.

3.3 Training and validation

The dev set is composed of 8,466 open class words, each accompanied by its context and associated PoS tags. Due to the significant size of the dev set, we made the decision to allocate only a small portion of it as a validation set. In particular, we set aside just 0.05 of the dev set to serve as the validation set, as a larger validation set would have required a larger amount of data for the training of our network. Despite its smaller size, the validation set provides an important measure of the model's accuracy and helps us fine-tune our model for improved performance.

The validation set and early stopping technique were used to identify the model with the best generalization capacity. By monitoring the performance of the model on the validation set during training, we were able to determine when the model had stopped improving and was at risk of overfitting to the training data. By using early stopping, we were able to prevent overfitting and select the model with the best generalization performance, ensuring that it would perform well on unseen data.

4. Ada Fine-tuning

GPT-3 models, developed by OpenAI, have gained fame in the field of NLP due to their remarkable ability to generate human-like text and perform a wide range of language-based tasks. These models have been trained on an extensive corpus of text data from the internet and possess an impressive capacity to understand and generate coherent and contextually relevant responses.

The Ada Model, developed by OpenAI, is an advanced and efficient language model within the GPT3.5 series. It stands out as the smallest and fastest model in this series. The exact parameter count of the Ada GPT model has not been publicly disclosed. However, based on its relation to GPT-3, it can be inferred that the Ada GPT model would have a similar scale with billions of parameters.

The OpenAI API[2] presents the option to fine-tune GPT3 models, and we chose to explore fine-tuning Ada for our lemmatization task. This approach allows us to enhance Ada's capabilities and tailor it specifically to handle accurate lemma identification and generation.

4.1 Prompt generation

Ada is capable of accepting natural language inputs, and for the purpose of fine-tuning, we need to convert our samples into prompts. This prompt-based approach allows us to customize Ada's functionality to suit our lemmatization requirements.

For fine-tuning Ada, we adopt a similar input format as described in the previous section. Ada itself takes care of tokenizing each sample. Therefore, we don't need to explicitly encode the data before inputting it.

For each sample in our dataset, we transform it into a JSONL object that contains two key-value pairs: prompt and completion. The prompt field holds the input, which includes the context, tags, and word. On the other hand, the completion field represents the desired output, which in our case is the lemma.

```
{"prompt":
    "Context: che spiega : \" ho preferito non innescare polemiche con le
        autorit&agrave; sanitarie .\n
    Tags: pron_rel v_gvrb p_oth p_oth v_avere v_pp adv v_gvrb nn prep art nn
        adj p_eos\n
    Word: sanitarie\n",
"completion":" sanitario\n"}
```

Since OpenAI does not provide an API for evaluating the model on a test set directly, we created a custom function that performs batch requests using the test set while ensuring compliance with the rate limits[3] imposed by the API. By carefully managing the requests, we can obtain valuable insights into the model's performance while adhering to the limitations imposed by the OpenAI API.

```python
predictions = []
batch_size = 20 # Max number of prompts per request
total_batches = len(prompts) // batch_size

for i in range(0, len(prompts), batch_size):
    # sleep for 1 second to avoid rate limit
    time.sleep(1)

    # Get batch of prompts
    prompt = prompts[i:i+batch_size]

    # Get predictions
    response = openai.Completion.create(
        model=model_id,
        prompt=prompt,
        max_tokens=20,
    )
    results = response.choices

    # Store predictions
    for j in range(len(results)):
        predictions.append(results[j].text)
```

The accuracy is calculated based on the number of correct predictions divided by the total number of predictions made by the Ada model. It provides an indication of how well the model performs on the given test dataset.

5. Result and discussion

The performance of our model will be evaluated using a test set shared among the participants of the EVALITA 2011 task, and will be compared against the results of other models.

5.1 Partecipants

These are the four participants in the EVALITA 2011 competition.

Name	**Institution**	**System Label**
Rodolfo Delmonte	University of Venice, Italy	Delmonte_UniVe
Djamè Seddah	Alpage (Inria)/Univ. Paris Sorbonne, France	Seddah_Inria-UniSorbonne
Maria Simi	University of Pisa, Italy	Simi_UniPi
Fabio Tamburini	University of Bologna, Italy	Tamburini_UniBo

Table 5.1: Partecipants

5.2 Open Class Words

In our evaluation of the model, we are focused solely on the accuracy of predicting the lemma of open-class words. Open-class words represent the most complex and interesting cases in natural language processing as they are constantly evolving and expanding, unlike closed-class or functional words that have a more limited number of words and do not change as often. Thus, evaluating the performance of the model in predicting the lemma of open-class words provides a more thorough understanding of its ability to disambiguate words in real-world text.

5.3 Evaluation

The tables below presents a comparison of accuracy, absolute error and relative error of our model with other participants in the Evalita2011 evaluation.

Table 5.2: Results

System	Lemmatisation Accuracy
Simi_UniPi	99.06%
Tamburini_UniBo	98.74%
Delmonte_UniVe	98.42%
Ada Model	**97.53%**
Our Model	**95.33%**
Seddah_Indria-UniSorbonne	94.76%

Table 5.3: Systems' absolute error distribution with respect to PoS-tags (computed as the error for each class divided by the total number of errors made by the system

System	**ADJ_***	**ADV**	**NN**	**V_***
Simi_UniPi	15.6%	8.2%	61.2%	15.0%
Tamburini_UniBo	17.7%	5.1%	64.4%	12.8%
Delmonte_UniVe	11.9%	6.7%	70.8%	10.6%
Ada Model	**26.2%**	**2.7%**	**39.8%**	**31.3%**
Our Model	**3.1%**	**2.9%**	**13.0%**	**81.1%**
Seddah_Indria-UniSorbonne	25.6%	4.9%	30.4%	44.1%

Table 5.4: Systems' relative error inside each lexical class (computed as the error made by the system for each class divided by the total number of token in the same class contained into the TS

System	ADJ_*	ADV	NN	V_*
Simi_UniPi	0.8%	0.7%	1.4%	0.5%
Tamburini_UniBo	1.2%	0.6%	2.0%	0.5%
Delmonte_UniVe	1.0%	1.0%	2.7%	0.6%
Ada Model	**3.7%**	**0.7%**	**2.6%**	**2.9%**
Our Model	**1.0%**	**1.7%**	**2.0%**	**17.3%**
Seddah_Indria-UniSorbonne	7.0%	2.4%	3.9%	8.1%

Compared to the other participants in EVALITA 2011, our system has demonstrated great performance in certain tag classifications, but has struggled with correctly lemmatizing verbs. This could be due to the limitations in the dataset used and the challenging nature of the Italian language, which has a multitude of complex verbal forms.

5.4 Conclusion

In conclusion, our model demonstrated a good level of accuracy when compared to the other participants in the Evalita2011 competition. While our model slightly outperformed the solution developed from Inria-UniSorbonne, which is based on a neural network, it still fell short in comparison to other participants who utilized more traditional solutions. It's important to note that one limitation of our model is the limited size of the development set, which may have impacted the final results. Nevertheless, this experiment highlights the potential for further improvement and optimization of our model in future iterations.

The Ada model has demonstrated impressive capabilities and has yielded good results in various tasks, but its fine-tuning may have been limited by a low number of epochs due to cost constraints. Increasing the number of epochs could potentially lead to even better results by allowing the model to learn more from the data.

Bibliography

[1] Fabio Tamburini. "The Lemmatisation Task at the EVALITA 2011 Evaluation Campaign". In: *Evaluation of Natural Language and Speech Tools for Italian*. Ed. by Bernardo Magnini et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 230–238. ISBN: 978-3-642-35828-9.

[2] *OpenAI API Reference*. `https://platform.openai.com/docs/api-reference/introduction`, note = Accessed: 2023-6-10.

[3] *OpenAI API Rate Limits*. `https://platform.openai.com/docs/guides/rate-limits`, note = Accessed: 2023-6-12.