# 

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Assignment 1: Word Vectors

# Training and Analysis

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Processament Oral i Escrit

# Curs 2024 - 2025

Grau en Ciència i Enginyeria de Dades

# Maite Blasi Garolera & Maria Sans Bosch

# Introduction

In this assignment, we aim to explore and improve the Continuous Bag-of-Words (CBOW) model used in word vector generation. The tasks involve working with a database derived from Wikipedia articles, and focusing on improving the CBOW model by introducing several modifications regarding weights for context words. Additionally, we will analyze the resulting word vectors and evaluate their effectiveness in identifying similar words, solving word analogies, etc.

# Task 1: Improve CBOW model

In the first part of the report, we explore the implementation and optimization of the **Continuous Bag of Words (CBOW) model.**

Traditionally, the CBOW model assigns equal weights to all context words when predicting a central word, which may limit its ability to capture more precise semantic relationships.

To address this limitation, we assign a different weight to each context word, either **fixed** or **learned** during training. Moreover, we will evaluate the impact of these modifications on the model performance, comparing **metrics such as accuracy and loss** under different configurations.

## Baseline: Uniform Weighting (Standard CBOW)

In the traditional CBOW model, all **context words contribute equally** to the prediction of the central word. This means that the context representation is computed as the simple **average of the word embeddings.**

## Fixed Position-Dependent Weighting

In this approach, we assign a **fixed set of weights** to the context words based on their relative position to the target word.

We used a symmetric weighting scheme: W= (1,2,3,3,2,1). In this way, we give **more importance to words closer to the target word** while reducing the influence of distant words.

As regards to this implementation, at first glance although it is simple and efficient to implement it is not the best option possible since **fixed weights may not generalize well across different datasets** and the predefined scheme may not be optimal for all tasks.

class CBOW(nn.Module):

def \_\_init\_\_(self, num\_embeddings, embedding\_dim):

super().\_\_init\_\_()

self.emb = nn.Embedding(num\_embeddings, embedding\_dim, padding\_idx=0)

self.lin = nn.Linear(embedding\_dim, num\_embeddings, bias=False)

self.register\_buffer('position\_weight', torch.tensor([1,2,3,3,2,1], dtype=torch.float32))

def forward(self, input):

e = self.emb(input)

weighted\_e= e\*self.position\_weight[:input.size(1)].view(1,-1,1)

u = weighted\_e.sum(dim=1)

z = self.lin(u)

return z



## b) Trained scalar weight for each position

Unlike the fixed scalar weighting, in this version, we allow the model to **learn scalar weights for each position** in the context window. This approach provides more flexibility, as the importance of each position can be adapted based on the training data.

class CBOW(nn.Module):

def \_\_init\_\_(self, num\_embeddings, embedding\_dim, context\_size):

super().\_\_init\_\_()

self.emb = nn.Embedding(num\_embeddings, embedding\_dim, padding\_idx=0)

self.lin = nn.Linear(embedding\_dim, num\_embeddings, bias=False)

self.position\_weight = nn.Parameter(torch.ones(context\_size, dtype=torch.float32))

def forward(self, input):

e = self.emb(input)

w = self.position\_weight.view(1,-1,1)

weighted\_e= e\*w

u = weighted\_e.sum(dim=1)

z = self.lin(u)

return z

...

context\_size=6

model = CBOW(len(vocab), params.embedding\_dim, context\_size).to(device)

print(model)

for name, param in model.named\_parameters():

print(f'{name:20} {param.numel()} {list(param.shape)}')

print(f'TOTAL {sum(p.numel() for p in model.parameters())}')



## c) Trained vector weight for each position

Lastly, we learn a **position-dependent weight vector** for each position in the context window. Each context word is element-wise multiplied by its corresponding position-dependent weight vector. The weighted vectors are then summed to predict the central word. This method allows for more complex interactions between the position of the context word and its contribution to the prediction.

Unlike the second methodology it is **more computationally expensive** and requires more training data to generalize well. However, we expect **much better results** since it allows us to capture contextual dependencies more effectively.

# Results

|  | Accuracy | | Loss | |
| --- | --- | --- | --- | --- |
|  | Wikipedia | El Periódico | Wikipedia | El Periódico |
| Standard | 23.9% | 15.6% | 5.12 | 6.11 |
| Fixed scalar weight | 31.4% | 21.2% | 4.56 | 5.61 |
| Trained scalar weight | 33.2% | 22.9% | 4.37 | 5.37 |
| Trained vector weight | 40.3% | 29.1% | 3.68 | 4.61 |

As we expected, the **Standard Model**, which assigns equals weights to all context words, has the lowest accuracy and highest loss. This indicates that a uniform weighting scheme is not appropriate for capturing word relationships in an effective way.

Once we assign a **different weighting based on the word position**, results on accuracy and loss improve. This suggests that giving more importance to closer context words helps the model learn better the relationships. However, because the weights are predefined, not learned, they may not be optimal across different datasets and tasks.

When **scalar weights are learned**, an improvement can also be seen with respect to the previous model implementations regarding accuracy and loss. By allowing the model to learn individual weights for each context position, it can dynamically **adjust word contributions**, leading to a better representation and more efficient configuration.

Finally, the **trained element-wise weight vector** provides the best performance. This is because each word in the window is multiplied element by element by its corresponding vector, not just by one scalar. This implementation allows modeling more complex interactions between words and their positions, but is also more expensive in terms of computation and the improvement with respect to the previous model is not significant.

Therefore, we can conclude that introducing **trainable parameters enhances accuracy while lowering the loss**. However, we have to take into account the **computational cost increases as we make the model more complex**.

Additionally, a better performance can be seen in the model based on the ‘Wikipedia’ dataset rather than the ‘El Periódico’ one. This is probably due to the fact that the first one contains more detailed information about the meaning and characteristics of the words, while the second one is based on periodistic texts which do not provide this type of information. Moreover, the training is done with the Wikipedia dataset, which makes the model learn more concrete characteristics about it.

# Task 2: Word vector Analysis

## a) Implement the WordVector class (Word Vector Analysis notebook) with the most\_similar and analogy methods to find closest vectors and analogies using the cosine similarity measure.

In order to implement the *most\_similar* method, we first check if the word from which we want to obtain the most similar words can be **found in the vocabulary** and get its index to obtain the vector representation. After that, we **calculate similarities** between this vector and the others. It finally **sorts the words based on this measure** and returns the first 10 and their corresponding similarities.

def most\_similar(self, word, topn=10):

if word not in self.word2idx:

return f"{word} not found in vocabulary"

word\_idx = self.word2idx[word]

word\_vector = self.vectors[word\_idx].reshape(1, -1)

similarities = cosine\_similarity(word\_vector, self.vectors)[0]

similar\_indices = similarities.argsort()[-topn-1:-1][::-1]

return [(self.idx2word[i], similarities[i]) for i in similar\_indices]

To make the implementation of the *analogy* method, we need to follow the next steps. First of all, we also **check whether all words are included in our vocabulary** and obtain the index corresponding to each one of them in case all are found on it to find the vectors representing them. Then, we **calculate the analogy vector** using the expression *analogy\_vector = vec\_x2 - vec\_x1 + vec\_y1* and use *cosine\_similarity* function to measure similarities between the words contained in the vocabulary and our analogy vector. Finally, we **sort the indexes of these words by decreasing number of similarities** and obtain the top 5 corresponding words (which are not the input ones).

def analogy(self, x1, x2, y1, topn=5):

if any(word not in self.word2idx for word in [x1, x2, y1]):

return "One or more words not found in vocabulary"

vec\_x1 = self.vectors[self.word2idx[x1]]

vec\_x2 = self.vectors[self.word2idx[x2]]

vec\_y1 = self.vectors[self.word2idx[y1]]

analogy\_vector = vec\_x2 - vec\_x1 + vec\_y1

similarities = cosine\_similarity(analogy\_vector.reshape(1, -1), self.vectors)[0]

sorted\_indices = similarities.argsort()[::-1]

results = [(self.idx2word[i], similarities[i]) for i in sorted\_indices]

results = [res for res in results if res[0] not in {x1, x2, y1}]

return results[:topn]

## 

## b) Intrinsic evaluation: perform an informal evaluation finding good and bad examples of closest words and analogies. You can analyze the behavior of the CBOW word vectors for words with multiple meanings, synonyms, and antonyms, word frequency, different types of analogies, bias (gender, race, sexual orientation, etc.).

**Similarities**

Regarding the word *hola*, since it is such a frequent word that can be used in multiple contexts alongside completely different words, we consider it important to highlight its output. The output of *hola* returns seemingly random words such as *addictiu, Frizer, Tori, Llucifer*, etc. This is likely due to the fact that this model has **learned associations within the Wikipedia corpus,** where *hola* appears in highly varied and different contexts.

The word *hola* often appears in informal texts or dialogues without a clear context, which may lead to its association with words that have no relation to each other. Likewise, ***hola* does not have a deep semantic meaning**, making it difficult to categorize within a specific semantic field.

In the example of the synonymous words *acabar* and *finalitzar*, we observe that in both cases, **the word with the highest similarity is the other one,** which demonstrates good behavior and aligns with expected results. Moreover, most of the words that are among the most similar (*continuar, reprendre, iniciar…*) coincide in both examples and, although they are not ranked in the same order for both words, they exhibit a fairly similar cosine similarity.

Regarding the word *clau*, we see that the algorithm has correctly learned which words are similar to it, such as *essencial, fonamental, crucial*, etc. However, all of them refer to the same meaning of the word without considering that it is polysemous. Therefore, it should also account for its other meanings, such as the metal piece used to open or close a door or the orthographic symbol used to enclose a set of numbers or letters, among others.

We observe that words like *depressió* contain **evident biases,** as their associations tend to **reinforce negative social stigmas** (with words such as *paranoia, pertorbació*) rather than focusing on more objective aspects of the condition, such as clinical symptoms or medical factors. This suggests a **need to refine the model to prevent perpetuating stereotypes or social stigmas** and to ensure a more neutral representation regarding topics like mental health.

Regarding gender, we see a very **evident and pronounced bias**, with highly **problematic and pejorative associations related to the word *dona*** (*prostituta*), as well as connections that reflect a traditional and outdated view of women's roles (*esposa, muller, parella*). On the other hand, some of the words found to be similar to *home* are *heroi, àngel* and *individu*, highlighting this difference. While *dona* is disparaged by being linked to words like *prostituta*, *home* is praised with the qualities shown and is given value as an individual, unlike *dona*. This tendency not only **reveals an implicit sexist bias in the data** but also underscores the **importance of refining the corpus** used for training the model and implementing strategies to mitigate these evident biases.

Finally, we have found words that are not in the vocabulary, as they are infrequent words, making it impossible to find similar words to them. Some examples include *xeròfit, baldraga, ginjoler*, or *xerigot*.

| **Pluja** | | **Hola** | |
| --- | --- | --- | --- |
| boira | 0.70734584 | addictiu | 0.42201132 |
| precipitació | 0.6354425 | Frízer | 0.4142839 |
| tempesta | 0.6280159 | Tori | 0.41038233 |
| neu | 0.60800576 | Llucifer | 0.40491015 |
| nevada | 0.59092605 | Fogars | 0.40468037 |
| foscor | 0.58727694 | Very | 0.4003104 |
| contaminació | 0.5737747 | Èol | 0.39734906 |
| sequera | 0.5694231 | Eris | 0.392035 |
| humitat | 0.56470406 | Moreu | 0.38933173 |
| calor | 0.5631335 | change | 0.38838774 |

| **Acabar** | | **Finalitzar** | |
| --- | --- | --- | --- |
| finalitzar | 0.85427374 | acabar | 0.85427374 |
| continuar | 0.7768503 | completar | 0.7348928 |
| reprendre | 0.72237915 | culminar | 0.71979797 |
| iniciar | 0.72219193 | concloure | 0.7188999 |
| seguir | 0.7212424 | iniciar | 0.69564235 |
| culminar | 0.69622827 | reprendre | 0.6933955 |
| concloure | 0.69462883 | prosseguir | 0.65387654 |
| prosseguir | 0.6849768 | continuar | 0.6306654 |
| completar | 0.6800364 | repetir | 0.6208492 |
| interrompre | 0.6551502 | reiniciar | 0.6115372 |

| **Clau** | | **Depressió** | |
| --- | --- | --- | --- |
| essencial | 0.62424016 | sequera | 0.66846657 |
| fonamental | 0.6054996 | crisi | 0.6519402 |
| crucial | 0.5874636 | malaltia | 0.6327848 |
| determinant | 0.569626 | solitud | 0.6230126 |
| decisiu | 0.5347402 | inestabilitat | 0.6136472 |
| recurrent | 0.49507117 | misèria | 0.6125088 |
| important | 0.4877603 | soledat | 0.6104492 |
| rellevant | 0.48581043 | paranoia | 0.6079661 |
| preliminars | 0.48518115 | pertorbació | 0.6048654 |
| individualització | 0.48142582 | virèmia | 0.5950274 |

| **Dona** | | **Home** | |
| --- | --- | --- | --- |
| noia | 0.72106194 | ancià | 0.6398561 |
| nena | 0.6401719 | heroi | 0.63224286 |
| prostituta | 0.6296749 | noi | 0.62976336 |
| esposa | 0.62486243 | àngel | 0.6291533 |
| criatura | 0.6152827 | individu | 0.6177017 |
| dama | 0.61004156 | infant | 0.6154979 |
| muller | 0.60097355 | animal | 0.6142865 |
| parella | 0.5897816 | nen | 0.5979688 |
| filla | 0.5881611 | esclau | 0.57801807 |
| tia | 0.5861123 | insult | 0.559536 |

**Analogies**

After trying different groups of words, we have found some examples which perform well and others which do not give the desired results:

| **dilluns, dimarts, dimecres**  **(dimarts - dilluns) + dimecres =** | | **dos, 2, tres**  **(2 - dos) + tres =** | |
| --- | --- | --- | --- |
| **dijous** | 0.95958924 | **3** | 0.9671368 |
| Elda | 0.9551065 | 5 | 0.96034384 |
| matinal | 0.95434654 | 6 | 0.95933473 |
| dissabtes | 0.9534545 | 4 | 0.9575169 |
| Som | 0.95342916 | 7 | 0.95644605 |
|  |  |  |  |
| **França, francès, Polònia**  **(francès - França) + Polònia =** | | **España, Madrid, França**  **(Madrid - Espanya) + França =** | |
| **polonès** | 0.945851 | **París** | 0.9273938 |
| rus | 0.94187474 | Londres | 0.9247048 |
| alemany | 0.9377252 | Roma | 0.91885364 |
| japonès | 0.9350022 | Berlín | 0.9141225 |
| neerlandès | 0.9333681 | Barcelona | 0.9111389 |

The examples above seem to **perform well**, since for all of them the word with the highest similarity score is the **one we expected to obtain.**

For the triplet belonging to days of the week, we rest the first day to the second, so when we add the third one, we expect to obtain the next day of the week, and indeed is the result we obtain. However, the other words which are given as a result of the analogy don’t seem to make sense since, except *dissabte*, they don’t have a clear relationship with the analogy we are analyzing.

In the next example with numbers, since we rest to the *2* its textual representation, we keep with the characteristic of being a digit, so when we add the name of number three, it makes sense that we obtain its corresponding digit in the first position and other digits in the following ones.

In the same way, subtracting *França* to *francès* makes the algorithm keep with the characteristic of being a language, then when adding the country *Poland* its corresponding language is obtained as the result with the highest similarity. Therefore, it seems like the model is learning the structure country-language well.

Finally, in the last example where we analyze the analogy between *España* and *Madrid* to find that their relation is the one of a country-capital, when we add the word *França*, its capital is given as a result of the implementation, and the next most similar words are also capitals of important countries. Then, **we can conclude that the algorithm performs well in this case.**

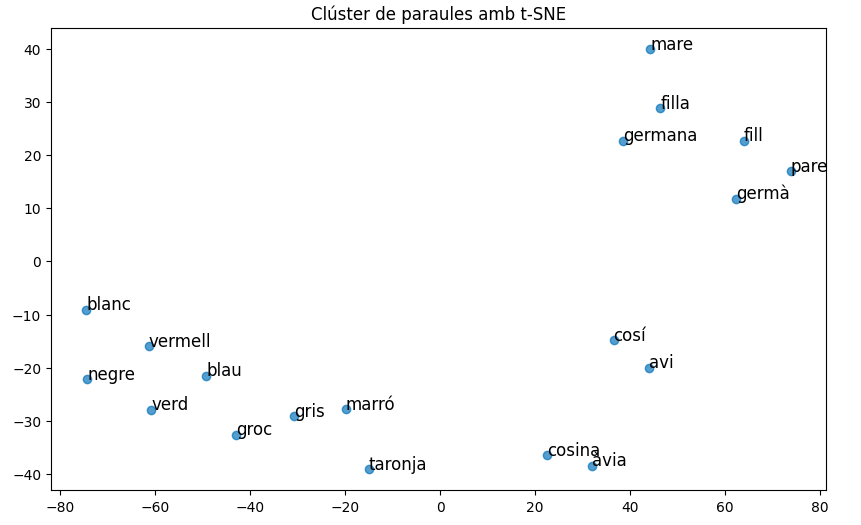
| **el, els, la**  **(els - el) + la =** | | **poma, taronja, pera**  **(taronja - poma) + pera =** | |
| --- | --- | --- | --- |
| dels | 0.91362906 | rodona | 0.94329906 |
| <unk> | 0.90441555 | verda | 0.942135 |
| una | 0.90391487 | porpra | 0.9381302 |
| també | 0.9016238 | blava | 0.9353842 |
| **les** | 0.90147144 | Hart | 0.9346556 |

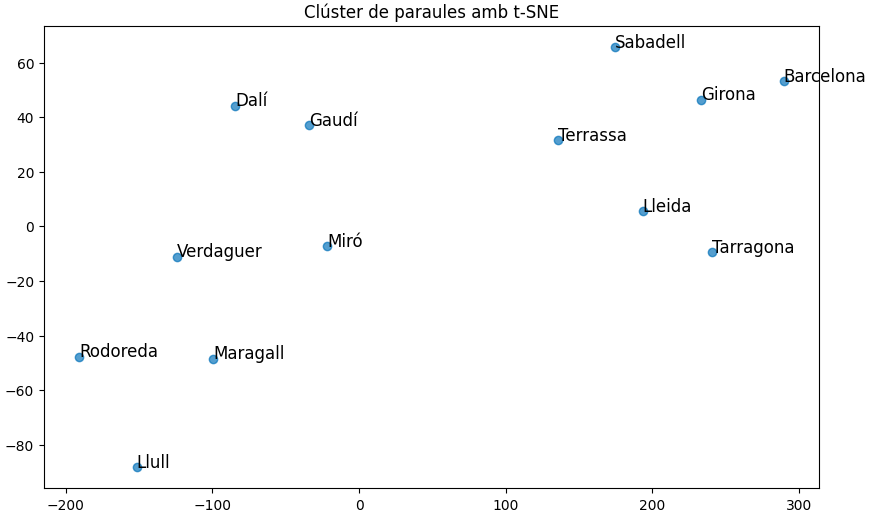
In the case of the articles *el, els, la* the model suggests the word *dels* as the best option, which is probably due to the way in which the words are distributed in the training data. Although it seems the model understands the relationship between *el* and *els* is singular-plural because the word les appears as the fifth option, we expected to obtain this solution in the first position.

However, in the next example it seems that the analogy is failing because, instead of finding other fruits, the model has prioritized words related to colors and shapes. It seems that the algorithm has learned that the differences between these fruits are more related to color than to their type of fruit.

## c) (Optional) Visualize word analogies or word clustering properties

For this task, we decided to **use the PCA and t-SNE algorithms** to **reduce the high-dimensional word vectors into two dimensions**. This enables us to visually explore how words cluster in the vector space and **analyze their relationships**, since we expect words which are related to be closer in the space where the embeddings are represented.



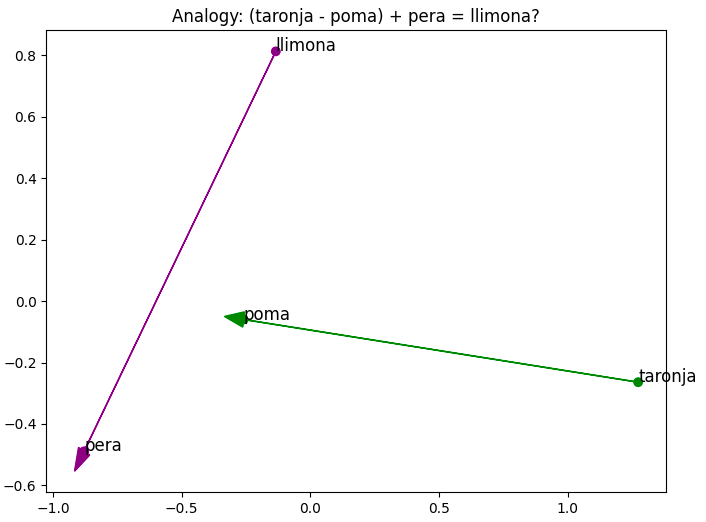
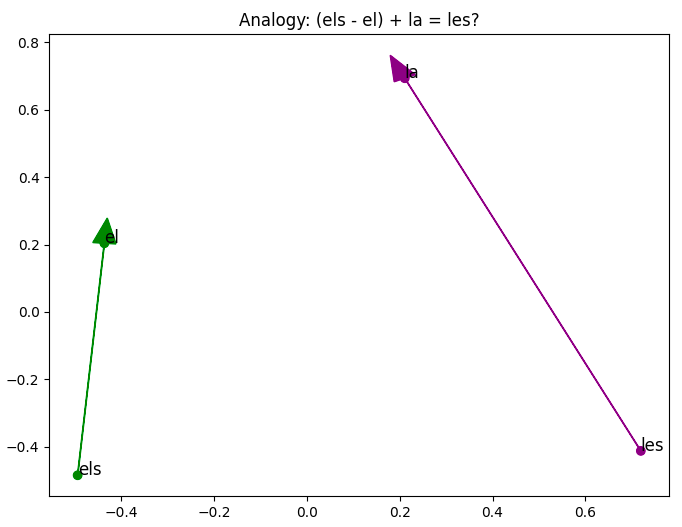
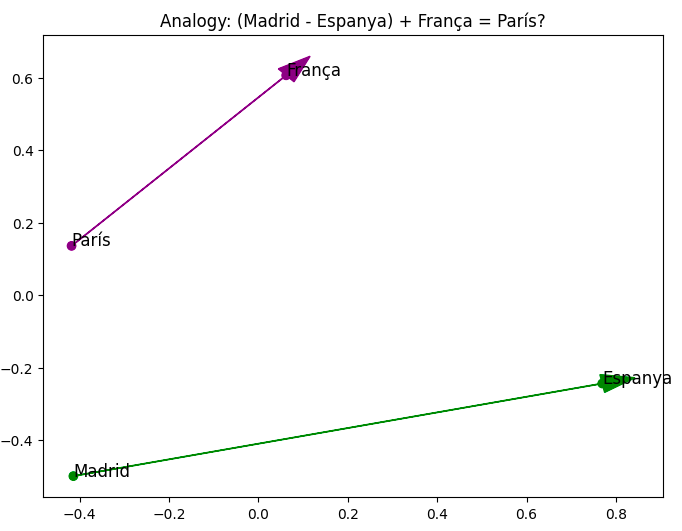
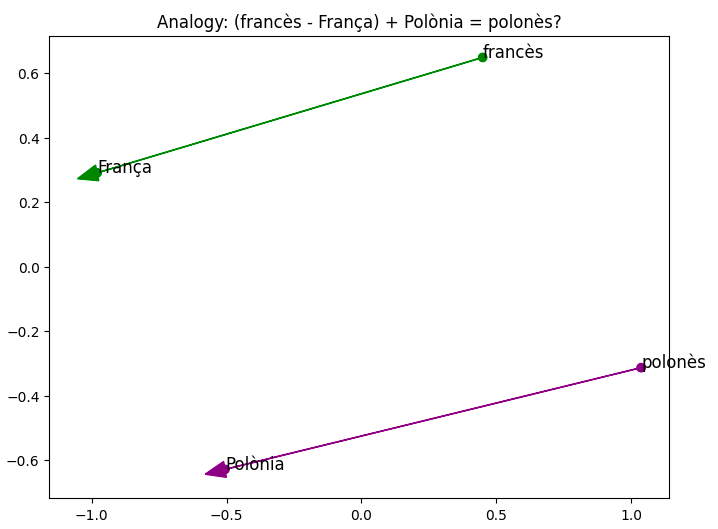
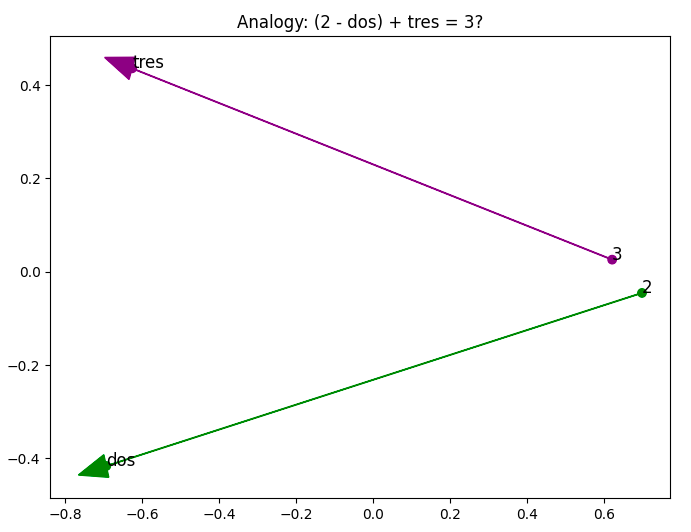
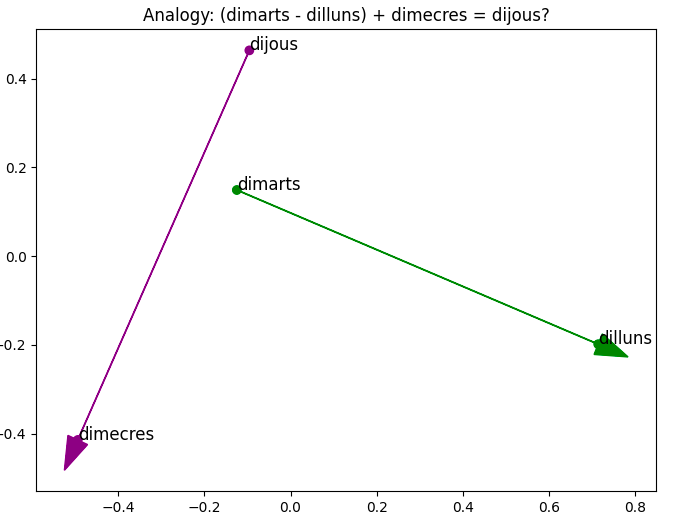


In order to be able to visualize the desired effect, we have chosen two examples where clusters can be differentiated properly.

In the first example, a clear separation can be seen between the **group of words which correspond to family relationships and types of colors.** Moreover, some clear analogies regarding genre can be seen between *cosina, cos*í and *àvia, avi*, for example.

We can also observe the same effect in the second example, where we obtain **one cluster for historical figures and another one for cities of Catalunya.**

Next, we have decided to plot the same analogies we have analyzed before in order to visualize their behavior in the 2-dimensional space.



From the analogies above, the one with the **best performance is the third one**, since the vectors which join the languages with the countries where they are spoken have the **same size, direction and are very close to being parallel.**

Although the analogies textual-numerical representations, country-capital and singular-plural don’t perform as well as the one mentioned before, they also show that the model is being **able to learn their grammatical and semantic relationships**, since the vectors have similar lengths and angles.

However, in the case where the model is trying to model the sequence of the days of the week and the relationship between fruits, the vectorial representation is not representing these in a way that the analogies are properly seen. The reason for this happening is probably because the PCA makes a dimensionality reduction, from a high number of dimensions to just two, with the objective of visualizing the analogy in a two dimensional space. Therefore, it **may not be capturing the relationship in the dimensions to which they are being reduced although the analogy is correctly found**, as it happens in the case of the days of the week.

In the case of the citric-not citric fruits, the problem seems to be the difficulty in finding the analogy, as we have seen before, more than in the representation, maybe because the relationship we are trying to find is not as evident as the model needs to be captured.

## d) (Optional) Prediction accuracy: compare the accuracy of the implemented CBOW models in the out-of-domain (el Periódico) test set (prediction of the central word given a context of 3 previous and 3 next words). To obtain your score on the competition test set you have to commit your version of the CBOW Training notebook, wait until the training completes, and then "Submit to competition" the obtained file (submission.csv) in the Output section of the notebook.

We have **submitted our version of the CBOW Training notebook**, waited for the training to complete and then submitted the obtained submission.csv file to the competition test set.

