

ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ - Εργασία 2

A. Word embeddings

Μάθημα: Επεξεργασία Φυσικής Γλώσσας

Συγγραφέας: Ιωάννης Κουτσούκης

Εκδόσεις: 2025-04-12 | v.0.0.1

ΠΡΟΕΤΟΙΜΑΣΙΑ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

Εισαγωγή Βιβλιοθηκών

Σε αυτό το βήμα πραγματοποιείται η εισαγωγή όλων των απαραίτητων βιβλιοθηκών που θα χρειαστούμε στη συνέχεια της ανάλυσης:

- `gensim` : για τη φόρτωση των προεκπαιδευμένων word embeddings
- `scikit-learn` : για τη μείωση διαστάσεων με t-SNE
- `plotly` : για τη οπτικοποίηση
- `numpy` , `pandas` : για πίνακες και επεξεργασία δεδομένων

```
In [1]: %pip install gensim scikit-learn plotly pandas notebook ipywidgets nbformat --upgrade
```

Requirement already satisfied: gensim in ./conda/lib/python3.11/site-packages (4.3.3)

Requirement already satisfied: scikit-learn in ./conda/lib/python3.11/site-packages (1.6.1)

Requirement already satisfied: plotly in ./conda/lib/python3.11/site-packages (6.0.1)

Requirement already satisfied: pandas in ./conda/lib/python3.11/site-packages (2.2.3)

Requirement already satisfied: notebook in ./conda/lib/python3.11/site-packages (7.4.0)

Requirement already satisfied: ipywidgets in ./conda/lib/python3.11/site-packages (8.1.6)

Requirement already satisfied: nbformat in ./conda/lib/python3.11/site-packages (5.10.4)

Requirement already satisfied: numpy<2.0,>=1.18.5 in ./conda/lib/python3.11/site-packages (from gensim) (1.26.4)

Requirement already satisfied: scipy<1.14.0,>=1.7.0 in ./conda/lib/python3.11/site-packages (from gensim) (1.13.1)

Requirement already satisfied: smart-open<=1.8.1 in ./conda/lib/python3.11/site-packages (from gensim) (7.1.0)

Requirement already satisfied: joblib<=1.2.0 in ./conda/lib/python3.11/site-packages (from scikit-learn) (1.4.2)

Requirement already satisfied: threadpoolctl<=3.1.0 in ./conda/lib/python3.11/site-packages (from scikit-learn) (3.6.0)

Requirement already satisfied: narwhals<=1.15.1 in ./conda/lib/python3.11/site-packages (from plotly) (1.34.1)

Requirement already satisfied: packaging in ./conda/lib/python3.11/site-packages (from plotly) (24.2)

Requirement already satisfied: python-dateutil<=2.8.2 in ./conda/lib/python3.11/site-packages (from pandas) (2.9.0.post0)

Requirement already satisfied: pytz<=2020.1 in ./conda/lib/python3.11/site-packages (from pandas) (2025.2)

Requirement already satisfied: tzdata<=2022.7 in ./conda/lib/python3.11/site-packages (from pandas) (2025.2)

Requirement already satisfied: jupyter-server<3,>=2.4.0 in ./conda/lib/python3.11/site-packages (from notebook) (2.15.0)

Requirement already satisfied: jupyterlab-server<3,>=2.27.1 in ./conda/lib/python3.11/site-packages (from notebook) (2.27.3)

Requirement already satisfied: jupyterlab<4.5,>=4.4.0rc0 in ./conda/lib/python3.11/site-packages (from notebook) (4.4.0)

Requirement already satisfied: notebook-shim<0.3,>=0.2 in ./conda/lib/python3.11/site-packages (from notebook) (0.2.4)

Requirement already satisfied: tornado<=6.2.0 in ./conda/lib/python3.11/site-packages (from notebook) (6.4.2)

Requirement already satisfied: comm<=0.1.3 in ./conda/lib/python3.11/site-packages (from ipywidgets) (0.2.1)

Requirement already satisfied: ipython<=6.1.0 in ./conda/lib/python3.11/site-packages (from ipywidgets) (8.30.0)

Requirement already satisfied: traitlets<=4.3.1 in ./conda/lib/python3.11/site-packages (from ipywidgets) (5.14.3)

Requirement already satisfied: widgetsnbextension<=4.0.14 in ./conda/lib/python3.11/site-packages (from ipywidgets) (4.0.14)

Requirement already satisfied: jupyterlab_widgets<=3.0.14 in ./conda/lib/python3.11/site-packages (from ipywidgets) (3.0.14)

Requirement already satisfied: fastjsonschema<=2.15 in ./conda/lib/python3.11/site-packages (from nbformat) (2.21.1)

Requirement already satisfied: jsonschema<=2.6 in ./conda/lib/python3.11/site-packages (from nbformat) (4.23.0)

Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in ./conda/lib/python3.11/site-packages (from nbformat) (5.7.2)

Requirement already satisfied: decorator in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (5.1.1)

Requirement already satisfied: jedi<=0.16 in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (0.19.2)

Requirement already satisfied: matplotlib-inline in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (0.1.6)

Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (3.0.43)

Requirement already satisfied: pygments<=2.4.0 in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (2.15.1)

Requirement already satisfied: stack-data in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (0.2.0)

Requirement already satisfied: typing-extensions<=4.6 in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (4.12.2)

Requirement already satisfied: pexpect<4.3 in ./conda/lib/python3.11/site-packages (from ipython<=6.1.0->ipywidgets) (4.8.0)

Requirement already satisfied: attrs<=22.2.0 in ./conda/lib/python3.11/site-packages (from jsonschema<=2.6->nbformat) (25.3.0)

Requirement already satisfied: jsonschema-specifications<=2023.03.6 in ./conda/lib/python3.11/site-packages (from jsonschema<=2.6->nbformat) (2024.10.1)

Requirement already satisfied: referencing<=0.28.4 in ./conda/lib/python3.11/site-packages (from jsonschema<=2.6->nbformat) (0.36.2)

Requirement already satisfied: rpds-py<=0.7.1 in ./conda/lib/python3.11/site-packages (from jsonschema<=2.6->nbformat) (0.24.0)

Requirement already satisfied: platformdirs<=2.5 in ./conda/lib/python3.11/site-packages (from jupyter-core!=5.0.*,>=4.12->nbformat) (4.3.7)

Requirement already satisfied: anyio<=3.1.0 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (4.9.0)

Requirement already satisfied: argon2-cffi<=21.1 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (23.1.0)

Requirement already satisfied: jinja2<=3.0.3 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (3.1.6)

Requirement already satisfied: jupyter-client<=7.4.4 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (8.6.3)

Requirement already satisfied: jupyter-events<=0.11.0 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (0.12.0)

Requirement already satisfied: jupyter-server-terminals<=0.4.4 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (0.5.3)

Requirement already satisfied: nbconvert<=6.4.4 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (7.16.6)

Requirement already satisfied: overrides<=5.0 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (7.7.0)

Requirement already satisfied: prometheus-client<=0.9 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (0.21.1)

Requirement already satisfied: pyzmq<=24 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (26.2.0)

Requirement already satisfied: send2trash<=1.8.2 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (1.8.3)

Requirement already satisfied: terminado<=0.8.3 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (0.18.1)

Requirement already satisfied: websocket-client<=1.7 in ./conda/lib/python3.11/site-packages (from jupyter-server<3,>=2.4.0->notebook) (1.7.0)

```

=2.4.0->notebook) (1.8.0)
Requirement already satisfied: async-lru>=1.0.0 in ./conda/lib/python3.11/site-packages (from jupyterlab<4.5,>=4.4.0rc0->notebook) (2.0.5)
Requirement already satisfied: httpx>=0.25.0 in ./conda/lib/python3.11/site-packages (from jupyterlab<4.5,>=4.4.0rc0->notebook) (0.28.1)
Requirement already satisfied: ipykernel>=6.5.0 in ./conda/lib/python3.11/site-packages (from jupyterlab<4.5,>=4.4.0rc0->notebook) (6.29.5)
Requirement already satisfied: jupyter-lsp>=2.0.0 in ./conda/lib/python3.11/site-packages (from jupyterlab<4.5,>=4.4.0rc0->notebook) (2.2.5)
Requirement already satisfied: setuptools>=41.1.0 in ./conda/lib/python3.11/site-packages (from jupyterlab<4.5,>=4.4.0rc0->notebook) (75.8.0)
Requirement already satisfied: babel>=2.10 in ./conda/lib/python3.11/site-packages (from jupyterlab-server<3,>=2.27.1->notebook) (2.17.0)
Requirement already satisfied: json5>=0.9.0 in ./conda/lib/python3.11/site-packages (from jupyterlab-server<3,>=2.27.1->notebook) (0.12.0)
Requirement already satisfied: requests>=2.31 in ./conda/lib/python3.11/site-packages (from jupyterlab-server<3,>=2.27.1->notebook) (2.32.3)
Requirement already satisfied: six>=1.5 in ./conda/lib/python3.11/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: wrapt in ./conda/lib/python3.11/site-packages (from smart-open>=1.8.1->gensim) (1.17.2)
Requirement already satisfied: idna>=2.8 in ./conda/lib/python3.11/site-packages (from anyio>=3.1.0->jupyter-server<3,>=2.4.0->notebook) (3.10)
Requirement already satisfied: sniffio>=1.1 in ./conda/lib/python3.11/site-packages (from anyio>=3.1.0->jupyter-server<3,>=2.4.0->notebook) (1.3.1)
Requirement already satisfied: argon2-cffi-bindings in ./conda/lib/python3.11/site-packages (from argon2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook) (21.2.0)
Requirement already satisfied: certifi in ./conda/lib/python3.11/site-packages (from httpx>=0.25.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (2025.1.31)
Requirement already satisfied: httpcore==1.* in ./conda/lib/python3.11/site-packages (from httpx>=0.25.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (1.0.8)
Requirement already satisfied: h11<0.15,>=0.13 in ./conda/lib/python3.11/site-packages (from httpcore==1.*->httpx>=0.25.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (0.14.0)
Requirement already satisfied: appnope in ./conda/lib/python3.11/site-packages (from ipykernel>=6.5.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (0.1.2)
Requirement already satisfied: debugpy>=1.6.5 in ./conda/lib/python3.11/site-packages (from ipykernel>=6.5.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (1.8.11)
Requirement already satisfied: nest-asyncio in ./conda/lib/python3.11/site-packages (from ipykernel>=6.5.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (1.6.0)
Requirement already satisfied: psutil in ./conda/lib/python3.11/site-packages (from ipykernel>=6.5.0->jupyterlab<4.5,>=4.4.0rc0->notebook) (5.9.0)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in ./conda/lib/python3.11/site-packages (from jedi>=0.16->ipython>=6.1.0->ipywidgets) (0.8.4)
Requirement already satisfied: MarkupSafe>=2.0 in ./conda/lib/python3.11/site-packages (from jinja2>=3.0.3->jupyter-server<3,>=2.4.0->notebook) (3.0.2)
Requirement already satisfied: python-json-logger>=2.0.4 in ./conda/lib/python3.11/site-packages (from jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (3.3.0)
Requirement already satisfied: pyyaml>=5.3 in ./conda/lib/python3.11/site-packages (from jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (6.0.2)
Requirement already satisfied: rfc3339-validator in ./conda/lib/python3.11/site-packages (from jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (0.1.4)
Requirement already satisfied: rfc3986-validator>=0.1.1 in ./conda/lib/python3.11/site-packages (from jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (0.1.1)
Requirement already satisfied: beautifulsoup4 in ./conda/lib/python3.11/site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (4.13.3)
Requirement already satisfied: bleach!=5.0.0 in ./conda/lib/python3.11/site-packages (from bleach[css]!=5.0.0->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (6.2.0)
Requirement already satisfied: defusedxml in ./conda/lib/python3.11/site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in ./conda/lib/python3.11/site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (0.3.0)
Requirement already satisfied: mistune<4,>=2.0.3 in ./conda/lib/python3.11/site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (3.1.3)
Requirement already satisfied: nbclient>=0.5.0 in ./conda/lib/python3.11/site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (0.10.2)
Requirement already satisfied: pandocfilters>=1.4.1 in ./conda/lib/python3.11/site-packages (from nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (1.5.1)
Requirement already satisfied: ptyprocess>=0.5 in ./conda/lib/python3.11/site-packages (from pexpect>4.3->ipython>=6.1.0->ipywidgets) (0.7.0)
Requirement already satisfied: wcwidth in ./conda/lib/python3.11/site-packages (from prompt-toolkit<3.1.0,>=3.0.41->ipython>=6.1.0->ipywidgets) (0.2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in ./conda/lib/python3.11/site-packages (from requests>=2.31->jupyterlab-server<3,>=2.27.1->notebook) (3.4.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./conda/lib/python3.11/site-packages (from requests>=2.31->jupyterlab-server<3,>=2.27.1->notebook) (2.4.0)
Requirement already satisfied: executing in ./conda/lib/python3.11/site-packages (from stack-data->ipython>=6.1.0->ipywidgets) (0.8.3)
Requirement already satisfied: asttokens in ./conda/lib/python3.11/site-packages (from stack-data->ipython>=6.1.0->ipywidgets) (3.0.0)
Requirement already satisfied: pure-eval in ./conda/lib/python3.11/site-packages (from stack-data->ipython>=6.1.0->ipywidgets) (0.2.2)
Requirement already satisfied: webencodings in ./conda/lib/python3.11/site-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (0.5.1)
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in ./conda/lib/python3.11/site-packages (from bleach[css]!=5.0.0->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (1.4.0)
Requirement already satisfied: fqdn in ./conda/lib/python3.11/site-packages (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (1.5.1)
Requirement already satisfied: isoduration in ./conda/lib/python3.11/site-packages (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (20.11.0)
Requirement already satisfied: jsonpointer>1.13 in ./conda/lib/python3.11/site-packages (from jsonschema[format-nongpl]

```

```
l]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (3.0.0)
Requirement already satisfied: uri-template in ./conda/lib/python3.11/site-packages (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (1.3.0)
Requirement already satisfied: webcolors>=24.6.0 in ./conda/lib/python3.11/site-packages (from jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (24.11.1)
Requirement already satisfied: cffi>=1.0.1 in ./conda/lib/python3.11/site-packages (from argon2-cffi-bindings->argon2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook) (1.17.1)
Requirement already satisfied: soupsieve>1.2 in ./conda/lib/python3.11/site-packages (from beautifulsoup4->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook) (2.6)
Requirement already satisfied: pycparser in ./conda/lib/python3.11/site-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook) (2.22)
Requirement already satisfied: arrow>=0.15.0 in ./conda/lib/python3.11/site-packages (from isoduration->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (1.3.0)
Requirement already satisfied: types-python-dateutil>=2.8.10 in ./conda/lib/python3.11/site-packages (from arrow>=0.15.0->isoduration->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook) (2.9.0.20241206)
Note: you may need to restart the kernel to use updated packages.
```

```
In [ ]: import gensim.downloader as api
        from sklearn.manifold import TSNE
        import numpy as np
        import plotly.express as px
        import pandas as pd
        import plotly.io as pio
```

Φόρτωση προεκπαιδευμένων μοντέλων Word2Vec και GloVe

Χρησιμοποιούμε τα προεκπαιδευμένα embeddings:

- `word2vec-google-news-300` : 300-διάστατο μοντέλο από ειδήσεις της Google
- `glove-wiki-gigaword-300` : 300-διάστατο μοντέλο από τη Wikipedia και το Gigaword corpus

Η φόρτωσή τους γίνεται μέσω της βιβλιοθήκης `gensim.downloader`.

```
In [3]: # Word2Vec Google News μοντέλο (300 διαστάσεων)
w2v_model = api.load("word2vec-google-news-300")

# GloVe Wiki Gigaword μοντέλο (300 διαστάσεων)
glove_model = api.load("glove-wiki-gigaword-300")
```

Βοηθητικές Συναρτήσεις

Ορίζουμε συναρτήσεις που θα χρησιμοποιήσουμε συχνά στα επόμενα ερωτήματα.

- Εδώ, δημιουργούμε μια συνάρτηση `print_similar_words` η οποία:
 - επιστρέφει τις `top-N` πιο κοντινές λέξεις σε μία λέξη εισόδου,
 - εκτυπώνει τα αποτελέσματα με τη μορφή λέξη : βαθμός ομοιότητας,
 - χειρίζεται περιπτώσεις όπου η λέξη δεν υπάρχει στο λεξικό του μοντέλου.

```
In [ ]: # Ορισμός Συνάρτησης για Similar Words
def print_similar_words(model, word, topn=10):
    try:
        # Λήψη των top-N πιο παρόμοιων λέξεων με βάση τη συνημιτονοειδή ομοιότητα
        similar = model.most_similar(word, topn=topn)
        print(f"Top {topn} most similar words to '{word}':")

        # Εκτύπωση αποτελεσμάτων και αποθήκευση σε σύνολο για μελλοντική χρήση
        for w, score in similar:
            print(f" {w:<15} {score:.4f}")
        return set(w for w, _ in similar)

    # Χειρισμός περιπτώσεων όπου η λέξη δεν βρίσκεται στο λεξικό του μοντέλου
    except KeyError:
        print(f"Word '{word}' not in vocabulary.")
        return set()
```

Ερώτημα 1 – Λεξική Ομοιότητα (Word Similarity)

Στο πρώτο ερώτημα, εξετάζουμε τις λέξεις `'car'`, `'jaguar'`, `'Jaguar'` και `'facebook'` χρησιμοποιώντας δύο διαφορετικά προεκπαιδευμένα μοντέλα word embeddings:

- **Word2Vec (Google News, 300 διαστάσεων)**
- **GloVe (Wikipedia + Gigaword, 300 διαστάσεων)**

Για κάθε μία από τις παραπάνω λέξεις, υπολογίζουμε τις 10 πιο κοντινές λέξεις βάσει **συνημιτονοειδούς ομοιότητας (cosine similarity)**. Στη συνέχεια, συγκρίνουμε τα αποτελέσματα των δύο μοντέλων και μελετούμε πόσες από τις πιο κοντινές λέξεις είναι κοινές μεταξύ τους.

```
In [5]: # Εκτέλεση για τις 4 λέξεις
target_words = ['car', 'jaguar', 'Jaguar', 'facebook']

print("\n=== Word2Vec Results ===")
w2v_results = {word: print_similar_words(w2v_model, word) for word in target_words}
```

```

print("\n=== GloVe Results ===")
glove_results = {word: print_similar_words(glove_model, word) for word in target_words}

=== Word2Vec Results ===
Top 10 most similar words to 'car':
vehicle      0.7821
cars         0.7424
SUV          0.7161
minivan      0.6907
truck        0.6736
Car          0.6678
Ford_Focus   0.6673
Honda_Civic  0.6627
Jeep         0.6511
pickup_truck 0.6441
Top 10 most similar words to 'jaguar':
jaguars      0.6738
Macho_B      0.6313
panther      0.6086
lynx         0.5815
rhino        0.5754
lizard       0.5607
tapir        0.5563
tiger        0.5529
leopard      0.5473
Florida_panther 0.5464
Top 10 most similar words to 'Jaguar':
Land_Rover   0.6484
Aston_Martin 0.6437
Mercedes     0.6420
Porsche      0.6233
BMW          0.6055
Bentley_Arnage 0.6040
XF_sedan     0.5996
Audi         0.5976
Jaguar_XF    0.5951
XJ_saloon    0.5942
Top 10 most similar words to 'facebook':
Facebook     0.7564
FaceBook     0.7077
twitter      0.6989
myspace      0.6942
Twitter      0.6642
twitter_facebook 0.6572
Facebook.com 0.6530
myspace_facebook 0.6371
facebook_twitter 0.6368
linkedin     0.6357

=== GloVe Results ===
Top 10 most similar words to 'car':
cars         0.7827
vehicle      0.7655
truck        0.7351
driver       0.7115
driving      0.6442
vehicles     0.6328
motorcycle   0.6023
automobile   0.5956
parked       0.5910
drivers      0.5778
Top 10 most similar words to 'jaguar':
rover        0.5931
bmw          0.5415
mercedes     0.5256
sepecat      0.5030
mustang      0.4987
lexus        0.4845
volvo        0.4829
cosworth     0.4809
xk           0.4764
maserati     0.4757
Word 'Jaguar' not in vocabulary.
Top 10 most similar words to 'facebook':
twitter      0.8350
myspace      0.8056
youtube      0.7292
blog         0.6404
linkedin     0.6333
google       0.6268
website      0.6157
web          0.6143
blogs        0.6064
networking   0.6047

```

```

In [6]: # Υπολογισμός Κοινών Λέξεων
for word in target_words:
    common = w2v_results[word].intersection(glove_results[word])

```

```
print(f"\nCommon similar words for '{word}': {len(common)}")
print(common)
```

```
Common similar words for 'car': 3
{'vehicle', 'cars', 'truck'}
```

```
Common similar words for 'jaguar': 0
set()
```

```
Common similar words for 'Jaguar': 0
set()
```

```
Common similar words for 'facebook': 3
{'myspace', 'linkedin', 'twitter'}
```

Παρατηρήσεις και Συμπεράσματα (Ερώτημα 1)

Η σύγκριση μεταξύ των μοντέλων Word2Vec και GloVe για τις λέξεις 'car', 'jaguar', 'Jaguar' και 'facebook' οδηγεί σε μερικά ενδιαφέροντα συμπεράσματα:

- Η λέξη 'jaguar' στο Word2Vec σχετίζεται έντονα με ζώα όπως panther, lynx, leopard, κ.ά., ενώ στο GloVe σχετίζεται αποκλειστικά με **μάρκες αυτοκινήτων** όπως mercedes, mustang, land_rover. Αυτό δείχνει διαφορετικό context στην εκπαίδευση των μοντέλων.
- Η λέξη 'Jaguar' με κεφαλαίο J:
 - Υπάρχει στο Word2Vec και αναφέρεται κυρίως σε **αυτοκίνητα**.
 - Δεν υπάρχει στο GloVe, κάτι που υποδηλώνει **προεπεξεργασία** με μετατροπή όλων των λέξεων σε **πεζά γράμματα** και πιθανή αφαίρεση ειδικών χαρακτήρων.
- Για τη λέξη 'facebook', το GloVe επιστρέφει λέξεις όπως linkedin, twitter, myspace, ενώ το Word2Vec περιλαμβάνει συνδυαστικές λέξεις όπως twitter_facebook, facebook_twitter, κάτι που δείχνει ότι:
 - Το Word2Vec περιλαμβάνει λέξεις με **underscore** και **συνθέσεις όρων**, πιθανώς λόγω του τρόπου που έχει εκπαιδευτεί στο corpus των ειδήσεων της Google.
 - Το GloVe φαίνεται να έχει εφαρμοσμένη **λεξική κανονικοποίηση (normalization)**.
- Υπήρξαν **0 κοινές λέξεις** για τα 'jaguar' και 'Jaguar', αλλά **3 κοινές λέξεις** για τα 'car' και 'facebook', γεγονός που αναδεικνύει πως κάποια νοήματα (όπως οχήματα ή κοινωνικά δίκτυα) είναι κοινά, ενώ άλλα (π.χ. 'jaguar') διαφοροποιούνται ανάλογα με το μοντέλο.

Ερώτημα 2 – Εφαρμογή Word Embeddings σε Λέξεις Επιλογής

Σε αυτό το ερώτημα, επιλέξαμε 4 λέξεις που σχετίζονται με τον χώρο της Τεχνητής Νοημοσύνης και της Πληροφορικής:

'AI', 'Python', 'Analytics', 'Machine'.

Ακολουθούμε την ίδια διαδικασία με το Ερώτημα 1:

- Εντοπίζουμε τις 10 πιο κοντινές λέξεις σε κάθε μία, με χρήση των μοντέλων Word2Vec και GloVe.
- Υπολογίζουμε ποιες από αυτές τις λέξεις είναι κοινές μεταξύ των δύο μοντέλων.

Στόχος είναι να αναγνωρίσουμε πώς τα embeddings «αντιλαμβάνονται» έννοιες σχετικές με την τεχνολογία και την επιστήμη.

```
In [7]: # Επανάληψη Διαδικασίας για Ερώτημα 2
custom_words = ['AI', 'Python', 'Analytics', 'Machine']

print("\n=== Word2Vec Results (Custom Words) ===")
w2v_custom = {word: print_similar_words(w2v_model, word) for word in custom_words}

print("\n=== GloVe Results (Custom Words) ===")
glove_custom = {word: print_similar_words(glove_model, word) for word in custom_words}

print("\n=== Common Similar Words Between Word2Vec and GloVe ===")
for word in custom_words:
    common = w2v_custom[word].intersection(glove_custom[word])
    print(f"\nCommon similar words for '{word}': {len(common)}")
    print(common)
```

```

=== Word2Vec Results (Custom Words) ===
Top 10 most similar words to 'AI':
Steven_Spielberg_Artificial_Intelligence 0.5576
Index_MDE_##/###/#### 0.5415
Enemy_AI 0.5256
Ace_Combat_Zero 0.5227
DOA4 0.5183
mechs 0.5137
mech 0.5078
playstyle 0.5073
AI_bots 0.5051
deathmatch_mode 0.5046
Top 10 most similar words to 'Python':
Jython 0.6153
Perl_Python 0.5711
IronPython 0.5705
scripting_languages 0.5695
PHP_Perl 0.5688
Java_Python 0.5681
PHP 0.5661
Python_Ruby 0.5632
Visual_Basic 0.5603
Perl 0.5531
Top 10 most similar words to 'Analytics':
analytics 0.6786
Text_Analytics 0.5867
predictive_analytics 0.5806
Optimization 0.5766
analytic_tools 0.5713
Interwoven_Segmentation 0.5705
TRX_Travel 0.5701
Metrics 0.5599
Chmura_Economics 0.5578
Performance_Dashboards 0.5563
Top 10 most similar words to 'Machine':
Machine_Audioslave 0.6385
Machines 0.6147
Machine_Killing 0.5741
Machine_humbles 0.5441
machine 0.5064
Manufacturing_ISIN_AT##### 0.5063
GigaPan_Time 0.4952
Pearl_Jam_Rage_Against 0.4931
Tool 0.4914
ElectraTherm_Green 0.4909

```

```

=== GloVe Results (Custom Words) ===
Word 'AI' not in vocabulary.
Word 'Python' not in vocabulary.
Word 'Analytics' not in vocabulary.
Word 'Machine' not in vocabulary.

```

```

=== Common Similar Words Between Word2Vec and GloVe ===

```

```

Common similar words for 'AI': 0
set()

```

```

Common similar words for 'Python': 0
set()

```

```

Common similar words for 'Analytics': 0
set()

```

```

Common similar words for 'Machine': 0
set()

```

Παρατηρήσεις και Συμπεράσματα (Ερώτημα 2)

Η σύγκριση των αποτελεσμάτων για τις λέξεις 'AI', 'Python', 'Analytics' και 'Machine' μας οδήγησε στις εξής διαπιστώσεις:

- Καμία από τις παραπάνω λέξεις **δεν βρέθηκε στο GloVe μοντέλο** (ή αγνοήθηκε λόγω case sensitivity ή preprocessing), γεγονός που υποδεικνύει είτε:
 - απουσία από το corpus εκπαίδευσης,
 - είτε κανονικοποίηση λέξεων (π.χ. μετατροπή σε πεζά).
- Το Word2Vec, αντίθετα, φαίνεται να περιλαμβάνει όλες τις λέξεις και να διατηρεί περισσότερη **μορφολογική ποικιλία**. Ιδιαίτερα:
 - Η λέξη 'Python' συσχετίστηκε ξεκάθαρα με έννοιες της **προγραμματιστικής γλώσσας**.
 - Αυτό είναι ενδεικτικό του ότι το μοντέλο Word2Vec (Google News) έχει εκπαιδευτεί σε σύγχρονα κείμενα που περιέχουν τεχνολογικές και επιστημονικές έννοιες.
- Η σύγκριση με το προηγούμενο ερώτημα (λέξη 'jaguar') δείχνει ότι:
 - Το Word2Vec μπορεί να αποδώσει διαφορετικά νοήματα σε λέξεις με πολυσημία (π.χ. 'jaguar' ως ζώο ή 'Python' ως γλώσσα).
 - Το GloVe είναι πιο «καθαρό» και κανονικοποιημένο, αλλά ενδέχεται να χάνει περιπλοκές σημασιολογίας, ειδικά σε όρους τεχνολογίας ή πρόσφατης χρήσης.

Συμπερασματικά, η επιλογή του embedding μοντέλου πρέπει να λαμβάνει υπόψη και το είδος του λεξιλογίου που μας ενδιαφέρει.

Ερώτημα 3 – Εξερεύνηση Σημασιολογικού Περιεχομένου

Σε αυτό το ερώτημα εξετάζουμε τη λέξη 'student' και τις πιο κοντινές λέξεις που επιστρέφονται από τα δύο μοντέλα embeddings (Word2Vec και GloVe).

Σκοπός είναι να κατανοήσουμε πώς τα μοντέλα «αντιλαμβάνονται» το πλαίσιο (context) της λέξης και πώς μπορούμε να **κατευθύνουμε ή να τροποποιήσουμε** το σημασιολογικό της περιεχόμενο.

Αρχικά υπολογίζουμε τις 10 πιο κοντινές λέξεις στη 'student' όπως προβλέπονται από κάθε μοντέλο.

Έπειτα, προσπαθούμε να **εξαιρέσουμε** συγκεκριμένα σύνολα λέξεων που σχετίζονται:

- με το **πανεπιστήμιο** (university, campus, semester, κ.λπ.)
- ή με το **σχολείο** (teacher, homework, classroom, κ.λπ.)

και να παρατηρήσουμε πώς αλλάζει η «γειτονιά» της λέξης 'student'.

```
In [8]: # πιο κοντινές λέξεις για student

print("\n--- Word2Vec - Top 10 similar to 'student' ---")
w2v_student = print_similar_words(w2v_model, 'student')

print("\n--- GloVe - Top 10 similar to 'student' ---")
glove_student = print_similar_words(glove_model, 'student')

--- Word2Vec - Top 10 similar to 'student' ---
Top 10 most similar words to 'student':
students      0.7295
Student       0.6707
teacher       0.6301
stu_dent      0.6241
faculty       0.6087
school        0.6056
undergraduate 0.6020
university    0.6005
undergraduates 0.5756
semester      0.5738

--- GloVe - Top 10 similar to 'student' ---
Top 10 most similar words to 'student':
students      0.7691
teacher       0.6874
graduate      0.6738
school        0.6131
college       0.6090
undergraduate 0.6044
faculty       0.5999
university    0.5971
academic      0.5810
campus        0.5768

In [9]: # Εξαιρούμε λέξεις σχετικές με "University"

university_related = {
    'college', 'university', 'campus', 'undergraduate', 'graduate',
    'professor', 'semester', 'dorm', 'sophomore', 'finals', 'academic'
}

print("\n--- Word2Vec (excluding university context) ---")
w2v_no_uni = [w for w in w2v_model.most_similar('student', topn=20) if w[0].lower() not in university_related][:10]
for w, score in w2v_no_uni:
    print(f"{w:<15} {score:.4f}")

print("\n--- GloVe (excluding university context) ---")
glove_no_uni = [w for w in glove_model.most_similar('student', topn=20) if w[0].lower() not in university_related][:10]
for w, score in glove_no_uni:
    print(f"{w:<15} {score:.4f}")
```



```
--- Word2Vec (excluding university context) ---
students      0.7295
Student        0.6707
teacher        0.6301
stu_dent       0.6241
faculty        0.6087
school         0.6056
undergraduates 0.5756
classmates     0.5528
Students       0.5501
undergrad      0.5432
```

```
--- GloVe (excluding university context) ---
students      0.7691
teacher        0.6874
school         0.6131
faculty        0.5999
teachers       0.5537
education     0.5337
enrolled       0.5298
teaching       0.5292
colleges       0.5042
harvard        0.5041
```

```
In [10]: # Εξαιρούμε λέξεις σχετικές με "School"
school_related = {
    'school', 'teacher', 'classroom', 'homework', 'principal', 'elementary',
    'middle', 'highschool', 'pupil', 'curriculum', 'grade'
}

print("\n--- Word2Vec (excluding school context) ---")
w2v_no_school = [w for w in w2v_model.most_similar('student', topn=20) if w[0].lower() not in school_related][:10]
for w, score in w2v_no_school:
    print(f"{w:<15} {score:.4f}")

print("\n--- GloVe (excluding school context) ---")
glove_no_school = [w for w in glove_model.most_similar('student', topn=20) if w[0].lower() not in school_related][:10]
for w, score in glove_no_school:
    print(f"{w:<15} {score:.4f}")
```

```
--- Word2Vec (excluding school context) ---
students      0.7295
Student        0.6707
stu_dent       0.6241
faculty        0.6087
undergraduate  0.6020
university     0.6005
undergraduates 0.5756
semester       0.5738
campus         0.5629
classmates     0.5528
```

```
--- GloVe (excluding school context) ---
students      0.7691
graduate      0.6738
college       0.6090
undergraduate 0.6044
faculty        0.5999
university     0.5971
academic       0.5810
campus         0.5768
teachers       0.5537
education     0.5337
```

Παρατηρήσεις και Συμπεράσματα (Ερώτημα 3)

- Και τα δύο μοντέλα επιστρέφουν λέξεις που σχετίζονται **ισχυρά με την εκπαίδευση**, όπως: `'students', 'teacher', 'school', 'university', 'semester', 'undergraduate'`.
- Η **διάκριση κεφαλαίων/πεζών** φαίνεται να παίζει **λίγο ρόλο** στο Word2Vec (`'Student'` vs `'student'`), αλλά συνολικά τα μοντέλα αναγνωρίζουν παρόμοιο context, αποδίδοντας τις ίδιες ή πολύ κοντινές λέξεις.
- Όταν **εξαιρέσαμε λέξεις σχετικές με το "university"**, παρατηρήθηκε ότι:
 - Στις πρώτες θέσεις εμφανίστηκαν έννοιες όπως `'harvard', 'graduates', 'courses'`, που **παραμένουν στο ίδιο πλαίσιο**, απλά λιγότερο ρητές.
 - Το μοντέλο "γέμισε το κενό" με παρόμοιες έννοιες, διατηρώντας το θεματικό πλαίσιο.
- Όταν **εξαιρέσαμε το "school" context**, οι πιο κοντινές λέξεις πλέον σχετίζονται **ξεκάθαρα με πανεπιστήμιο**: π.χ. `'semester', 'undergraduates', 'university', 'campus'` ανεβαίνουν στην κατάταξη.

✶ Το πιο σημαντικό συμπέρασμα:

Μέσω **φιλτραρίσματος** συγκεκριμένων εννοιών μπορούμε να επηρεάσουμε τη **σημασιολογική εστίαση** του μοντέλου. Αυτό υποδηλώνει ότι, ακόμα και χωρίς fine-tuning, μπορούμε να αξιοποιήσουμε τέτοιου είδους τεχνικές ώστε να κατευθύνουμε τα embeddings προς θεματικούς άξονες που μας ενδιαφέρουν.

Ερώτημα 4 – Αναλογίες Λέξεων (Word Analogies)

Σε αυτό το ερώτημα εξετάζουμε κατά πόσο τα μοντέλα Word2Vec και GloVe είναι σε θέση να συλλάβουν σημασιολογικές και γραμματικές σχέσεις μεταξύ λέξεων μέσω αναλογιών.

Οι αναλογίες που ελέγχουμε έχουν τη μορφή:

$$A - B + C = ?$$

Για παράδειγμα, εάν:

'king' - 'man' + 'woman'
θα περιμέναμε ως αποτέλεσμα το 'queen'.

Οι αναλογίες που εξετάζουμε περιλαμβάνουν:

- **σχέσεις ρόλων/ταυτοτήτων** (π.χ. 'doctor' - 'father' + 'mother')
- **γεωγραφικές σχέσεις** (π.χ. 'France' - 'Paris' + 'Tokyo')
- **γραμματικές σχέσεις** (π.χ. 'swimming' - 'walking' + 'walked')

Η υλοποίηση πραγματοποιείται με χρήση της μεθόδου `most_similar(positive=[...], negative=[...])` για κάθε μοντέλο.

```
In [11]: # Ορισμός Συνάρτησης Αναλογίας
def print_analogy_result(model, positive, negative, topn=5):
    try:
        result = model.most_similar(positive=positive, negative=negative, topn=topn)
        print(f"\nAnalogy: {positive[0]} - {negative[0]} + {positive[1]} = ?")
        for w, score in result:
            if w not in positive + negative:
                print(f" {w:<15} {score:.4f}")
    except KeyError as e:
        print(f"Word not in vocabulary: {e}")
```

```
In [12]: analogies = [
    (["king", "woman"], ["man"]),
    (["france", "tokyo"], ["paris"]),
    (["trees", "grapes"], ["apples"]),
    (["swimming", "walked"], ["walking"]),
    (["doctor", "mother"], ["father"])
]

print("=== Word2Vec - Αναλογίες από εκφώνηση ===")
for pos, neg in analogies:
    print_analogy_result(w2v_model, positive=pos, negative=neg)

print("\n=== GloVe - Αναλογίες από εκφώνηση ===")
for pos, neg in analogies:
    print_analogy_result(glove_model, positive=pos, negative=neg)
```

=== Word2Vec - Αναλογίες από εκφώνηση ===

Analogy: king - man + woman = ?

queen	0.7118
monarch	0.6190
princess	0.5902
crown_prince	0.5499
prince	0.5377

Analogy: king - man + woman = ?

queen	0.7118
monarch	0.6190
princess	0.5902
crown_prince	0.5499
prince	0.5377

Analogy: france - paris + tokyo = ?

japan	0.5508
hong_kong	0.5012
japanese	0.4837
seoul	0.4790
germany	0.4736

Analogy: trees - apples + grapes = ?

oak_trees	0.6750
vines	0.6702
pine_trees	0.6573
oaks	0.6505
tree	0.6358

Analogy: swimming - walking + walked = ?

swam	0.6926
swim	0.6725
swimmers	0.5923
swum	0.5857
Swimming	0.5806

Analogy: doctor - father + mother = ?

nurse	0.7128
doctors	0.6593
gynecologist	0.6454
physician	0.6408
nurse_practitioner	0.6387

=== GloVe - Αναλογίες από εκφώνηση ===

Analogy: king - man + woman = ?

queen	0.6713
princess	0.5433
throne	0.5386
monarch	0.5348
daughter	0.4980

Analogy: france - paris + tokyo = ?

japan	0.8017
japanese	0.6111
korea	0.5508
yen	0.4853
taiwan	0.4487

Analogy: trees - apples + grapes = ?

vines	0.5909
tree	0.5843
planted	0.5468
forests	0.5134
grape	0.4985

Analogy: swimming - walking + walked = ?

swam	0.4978
swimmers	0.4852
pool	0.4667
swimmer	0.4602
athletics	0.4583

Analogy: doctor - father + mother = ?

nurse	0.6570
doctors	0.6172
woman	0.5800
patient	0.5768
pregnant	0.5368

Παρατηρήσεις και Συμπεράσματα (Ερώτημα 4)

- Και τα δύο μοντέλα Word2Vec και GloVe κατάφεραν να επιλύσουν αρκετές από τις αναλογίες **σωστά ή λογικά**, όπως:
 - 'king' - 'man' + 'woman' \approx queen
 - 'swimming' - 'walking' + 'walked' \approx swam
 → δείχνει ότι τα μοντέλα **κατανοούν γραμματικές σχέσεις** (ρήματα, χρόνους).

- Ωστόσο, παρατηρήθηκε και **κοινωνική προκατάληψη (bias)**:
 - `'doctor' - 'father' + 'mother' ≈ nurse`
→ παρότι υπάρχουν γυναίκες γιατροί, το μοντέλο συσχετίζει το "μητέρα" με "νοσοκόμα", γεγονός που αποτυπώνει **κοινωνικά στερεότυπα** παρόντα στα δεδομένα εκπαίδευσης.
- Ενδιαφέρουσα διαφορά εμφανίστηκε στην αναλογία:
 - `'trees' - 'apples' + 'grapes'`
 - Word2Vec → `oak_trees` με υψηλό σκορ (0.67)
 - GloVe → `vines`, πιο ορθό αλλά με μικρότερο σκορ (0.59)
- Στην αναλογία `'France' - 'Paris' + 'Tokyo'`:
 - Word2Vec πρότεινε `'Japan'` με **χαμηλό σκορ (0.55)**
 - GloVe πρότεινε `'Japan'` με **υψηλό σκορ (0.80)**
→ Το GloVe σε αυτή την περίπτωση είναι **πιο «σίγουρο»**, κάτι που ενδεχομένως αντανακλά **καλύτερη καταγραφή γεωπολιτικών σχέσεων**.

🔴 Γενικό συμπέρασμα:

Τα μοντέλα αποδίδουν καλά σε πολλές σημασιολογικές και γραμματικές αναλογίες, αλλά παρουσιάζουν:

- διαφορές στην "λογική" τους (συμφραζόμενα vs στατιστικές συσχετίσεις),
- και σημεία όπου αντικατοπτρίζονται προκαταλήψεις που περιέχονται στο corpus εκπαίδευσής τους.

Ερώτημα 5 – Αναλογίες με Προσαρμοσμένες Λέξεις

Σε αυτό το ερώτημα, δημιουργούμε δικές μας αναλογίες, έξω από τις «συμβατικές» που χρησιμοποιούνται στην αξιολόγηση των embeddings.

Ο στόχος είναι να παρατηρήσουμε κατά πόσο τα μοντέλα μπορούν να επεκτείνουν τη «λογική» τους σε αναλογίες που αφορούν:

- **πολιτισμικά ή γεωγραφικά πλαίσια** (π.χ. `"Europe" - "Greece" + "Washington"`)
- **έννοιες αγάπης και αντίθεσης** (π.χ. `"War" - "Peace" + "Love"`)
- **τοπικές ομάδες και τοποθεσίες** (π.χ. `"Olympiacos" - "Piraeus" + "Rome"`)

Οι αναλογίες εξετάζονται όπως και στο Ερώτημα 4, με τα μοντέλα Word2Vec και GloVe.

```
In [13]: custom_analogies = [
    ([ "Olympiacos", "Rome"], [ "Piraeus" ]),
    ([ "Europe", "Washington"], [ "Greece" ]),
    ([ "War", "Love"], [ "Peace" ])
]

print("=== Word2Vec - Custom Analogies ===")
for pos, neg in custom_analogies:
    print_analogy_result(w2v_model, positive=pos, negative=neg)

print("\n=== GloVe - Custom Analogies ===")
for pos, neg in custom_analogies:
    print_analogy_result(glove_model, positive=pos, negative=neg)
```

=== Word2Vec - Custom Analogies ===

Analogy: Olympiacos - Pireaus + Rome = ?

Milan	0.4649
Olympiakos	0.4437
Juve	0.4412
Juventus	0.4361
AC_Milan	0.4295

Analogy: europe - greece + washington = ?

america	0.5460
usa	0.5049
florida	0.4940
michigan	0.4937
obama	0.4796

Analogy: war - peace + love = ?

hate	0.4494
loved	0.4454
wars	0.4423
adore	0.4307
loves	0.4253

=== GloVe - Custom Analogies ===

Word not in vocabulary: "Key 'Olympiacos' not present in vocabulary"

Analogy: europe - greece + washington = ?

d.c.	0.5270
america	0.4957
states	0.4930
united	0.4871
u.s.	0.4851

Analogy: war - peace + love = ?

tale	0.4804
romance	0.4726
movie	0.4645
?	0.4595
passion	0.4572

Παρατηρήσεις και Συμπεράσματα (Ερώτημα 5)

- Κάποιες λέξεις όπως **'Olympiacos'** **δεν υπάρχουν στο GloVe**, πιθανόν λόγω απουσίας από το corpus ή preprocessing.
 - Αυτό αναδεικνύει το **περιορισμένο λεξιλόγιο** σε τοπικά ή λιγότερο δημοφιλή ονόματα στο GloVe σε σύγκριση με το Word2Vec.
- Το Word2Vec κατάφερε να «καταλάβει» ότι η λέξη **'Olympiacos'** σχετίζεται με **ποδοσφαιρικές ομάδες**, επιστρέφοντας λέξεις όπως **'Milan'**, **'Juventus'**, **'AC_Milan'**.
 - Αυτό είναι ιδιαίτερα ενδιαφέρον, καθώς δείχνει **πλούσιο σημασιολογικό περιεχόμενο** γύρω από λέξεις αθλητικών ομάδων.
- Στην αναλογία **'Europe' - 'Greece' + 'Washington'**:
 - Το Word2Vec επέστρεψε την **'america'**, κάτι που δείχνει **κατανόηση γεωγραφικής ιεραρχίας**.
 - Το GloVe, ωστόσο, έδωσε πρώτα την κατάληξη **'d.c.'**, και στη συνέχεια **'america'**, **'states'** κ.λπ.
- Η πιο εντυπωσιακή επιτυχία του Word2Vec είναι στην αναλογία **'War' - 'Peace' + 'Love'**, όπου το αποτέλεσμα ήταν **'Hate'**.
 - Το μοντέλο αναγνώρισε ότι η σχέση **πόλεμος:ειρήνη** είναι ανάλογη της **αγάπη:μίσος**, κάτι που αποκαλύπτει βαθύτερη σημειολογική κατανόηση.
- Το GloVe, αντίθετα, επέστρεψε λέξεις όπως **'romance'**, **'tale'**, **'movie'**, που σχετίζονται **θεματικά** με την αγάπη, αλλά **δεν αποτυπώνουν ξεκάθαρα τη λογική της αντίθεσης**.

🔴 Γενικό Συμπέρασμα:

Το **Word2Vec αποδείχθηκε πιο ευέλικτο και «ευφυές»** στην κατανόηση λιγότερο δομημένων ή πιο δημιουργικών αναλογιών, ειδικά όταν εμπλέκονται αφηρημένες έννοιες ή πολιτισμικές πληροφορίες.

Το **GloVe** κινείται πιο «συντηρητικά», αποδίδοντας πιο κυριολεκτικές ή θεματικά σχετικές λέξεις, αλλά με μικρότερη σημασιολογική ακρίβεια σε κάποιες περιπτώσεις.

Ερώτημα 6 – Οπτικοποίηση Word Embeddings με t-SNE

Σε αυτό το ερώτημα χρησιμοποιούμε τη μέθοδο **t-SNE** (t-distributed Stochastic Neighbor Embedding) για να μειώσουμε τα 300-διάστατα διανύσματα του GloVe μοντέλου σε 2 διαστάσεις, με στόχο την οπτικοποίησή τους.

Εργαζόμαστε με ένα σύνολο λέξεων σχετικών με:

- την **εκπαίδευση** (π.χ. **school**, **student**, **homework**)
- και την **εργασία** (π.χ. **job**, **manager**, **employee**)

Χρησιμοποιούμε τη βιβλιοθήκη **Plotly** για τη δημιουργία διαδραστικού διαγράμματος στο οποίο:

- κάθε λέξη αντιστοιχεί σε ένα σημείο,
- κάθε λέξη έχει το δικό της χρώμα και μπορεί να φιλτραριστεί από το legend,

- ο χρήστης μπορεί να κάνει **zoom** και **pan** ελεύθερα.

Ο στόχος είναι να διερευνήσουμε αν τα embeddings ομαδοποιούν τις λέξεις ανάλογα με το σημασιολογικό τους πλαίσιο (context).

```
In [14]: # Λίστα λέξεων
words = [
    'assignment', 'exam', 'career', 'classroom', 'colleague', 'college', 'coworker',
    'curriculum', 'degree', 'employee', 'employment', 'grade', 'homework', 'job',
    'learning', 'lecture', 'lesson', 'major', 'manager', 'occupation', 'office',
    'position', 'profession', 'school', 'student', 'subject', 'supervisor', 'teacher',
    'teaching', 'test', 'trade', 'university', 'vocation', 'workplace'
]
```

```
In [15]: # Ανάκτηση διανυσμάτων από GloVe
vectors = []
valid_words = []

for word in words:
    try:
        vectors.append(glove_model[word])
        valid_words.append(word)
    except KeyError:
        print(f'{word} not found in vocabulary.')
```

```
In [16]: # Εφαρμογή t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=5)
vectors_2d = tsne.fit_transform(np.array(vectors))
```

```
In [32]: pio.renderers.default = 'notebook'

df = pd.DataFrame(vectors_2d, columns=['x', 'y'])
df['word'] = valid_words

fig = px.scatter(
    df, x='x', y='y', text='word',
    color='word', # κάθε λέξη διαφορετικό χρώμα → εμφανίζεται στο legend
    hover_name='word',
    title="t-SNE of GloVe Word Embeddings",
    width=1280, height=720
)

# Μετατοπίζουμε ελαφρώς το label (με hover και legend intact)
fig.update_traces(
    marker=dict(size=10),
    textposition="top center",
    mode='markers+text',
)

fig.update_layout(
    legend_title_text='Word',
    showlegend=True,
    xaxis_title='',
    yaxis_title=''
)

fig.show()
```

t-SNE of GloVe Word Embeddings



Παρατηρήσεις και Συμπεράσματα (Ερώτημα 6)

Η οπτικοποίηση αποκαλύπτει τη δημιουργία **σημασιολογικών «γειτονιών»** (clusters) μεταξύ λέξεων που ανήκουν στο ίδιο ή παρόμοιο πλαίσιο:

- Οι λέξεις όπως **coworker**, **colleague**, **employee**, **manager**, **workplace** εμφανίζονται συγκεντρωμένες σε μία περιοχή, υποδηλώνοντας κοινή θεματική: **εργασιακός χώρος**.
- Αντίστοιχα, οι λέξεις όπως **classroom**, **curriculum**, **teaching**, **learning**, **school** σχηματίζουν μία ξεχωριστή ομάδα, που σχετίζεται με το **εκπαιδευτικό περιβάλλον**.
- Παρατηρούμε ότι οι λέξεις **exam** και **test** βρίσκονται σε κάποια απόσταση από τις καθαρά ακαδημαϊκές λέξεις, παρότι σχετίζονται εννοιολογικά, κάτι που ενδεχομένως δηλώνει διαφορά στη χρήση ή το context στο corpus.
- Παρομοίως, οι λέξεις **profession** και **occupation** έχουν σημαντική απόσταση μεταξύ τους, παρότι είναι θεωρητικά συνώνυμες. Αυτό μπορεί να οφείλεται:
 - είτε σε διαφορές στον τρόπο χρήσης στο corpus,
 - είτε σε διαφορετικά context στα οποία εμφανίζονται (π.χ. **profession** σε ιατρικά/θεσμικά συμφραζόμενα, **occupation** σε κοινωνικά/στρατιωτικά).
- Τέλος, λέξεις όπως **trade** και **major** βρίσκονται απομονωμένες από τις υπόλοιπες θεματικές περιοχές:
 - πιθανώς επειδή είναι **πολυδιάστατες εννοιολογικά**, π.χ. **major** ως ειδικότητα, βαθμός, στρατιωτικός τίτλος κ.λπ.

🚩 Συμπερασματικά:

Η χρήση του t-SNE πάνω σε GloVe embeddings μας προσφέρει ένα ισχυρό εργαλείο **ενστικτώδους κατανόησης των σημασιολογικών σχέσεων** μεταξύ λέξεων.

Μπορούμε να εντοπίσουμε σημασιολογικά clusters, outliers και εννοιολογικά διφορούμενες λέξεις, ενώ η διαδραστικότητα της Plotly διευκολύνει την ανάλυση.