Zanjan University

Name: Reza Jafari **NLP Assignment 2** **Professor: Dr.Leila Safari**

Homework 2-1: Adventures with Word Embeddings

The aim of this assignment is to furnish you with a solid practical and theoretical understanding of the inner workings of word embeddings. In the first part of the assignment, you will explore the effects of the various hyperparameters in word embedding algorithms. In the second part, you will use word embeddings in a classification task.

General instructions

You should budget at least a day just for your full set of experiments to run successfully. This likely means several days' worth of runtime in the debugging phase. We strongly recommend that you start coding as soon as possible, but at minimum a week in advance of the deadline.

1 Parameter Search

The hyperparameters you will be exploring are:

- Architecture: (CBOW, Skip-gram)
- Dimension: (50, 100, 300)
- Window Size: (5, 10)
- Epochs: 5 (Train all models for exactly 5 epochs)
- *Note:* Keep negative sampling fixed at 5 to reduce search space.

This results in 12 unique settings (2*3 * 2).

Evaluation Implementation:

You will not be provided with evaluation scripts. Instead, you must implement the evaluation loop yourself using Gensim's built-in methods:

- Use evaluate_word_pairs() for the **WordSim353** task.
- Use evaluate_word_analogies() for the **BATS** tasks.
- **Note:** You are responsible for downloading the standard **WordSim353** and **BATS** datasets (available on typical NLP repositories like GitHub or Kaggle) and loading them correctly for these functions.

Writeup

Your submission for this part of the assignment will consist of

(1) A table containing the results of your parameter search;

(2) A written analysis of your results.

Each row of the table should contain numerical results for one choice of algorithm and parameter setting. The columns should include algorithm, context window, dimension, number of negative samples, correlation on WordSim353, accuracy for four BATS categories (you pick the ones you think are most interesting from among the 9 low-level categories, the 4 high-level categories, or the total score), and accuracy on the win353 paraphrase corpus. The table should look something like this:

Architecture	Win.	Dim.	#Neg.samples	WordSim	BATS 1	BATS 2	BATS 3	Win353-Para
Skip-gram	5	100	5	47.05	0.01	0.02	0.03	0.01
…	…	…	…	…	…	…	…	

Table 1: An example results table.

Your writeup should at least address the following prompts, but you are also encouraged to include other interesting observations or hypotheses you have made.

1. Does larger dimensionality always equate to better performance? In which categories and for which models? Why do you think this is?
2. Does better performance on one task mean better performance on the others? Provide a hypothesis as to why or why not.
3. Was performance roughly similar across all analogy categories? If different, how did it vary? Why do you think you observed this variation? Perform a brief error analysis and compare errors across the BATS categories you selected for your table.

Qualitative Analysis In addition to the numerical analysis, you must perform a qualitative inspection to understand what the model is actually learning:

- Pick **3 polysemous words** (words with multiple meanings), such as "bank", "apple", or "run".
- Find the **5 nearest neighbors** for these words in your **best performing model** vs. the **worst performing model** from Part 1.
- **Analyze the Window Size Effect:** Compare the neighbors. How did the context window size affect the semantic vs. syntactic similarity? (e.g., Does a small window like window=2 yield "running" for "run", while a larger window like window=10 yields "marathon"?).
- Report these findings in a table and briefly discuss them.

1.1 Training corpus

Instead of the classic Brown corpus, you will train your embedding models on the WikiText-2 dataset (or text8). This dataset allows for capturing richer semantic relationships due to its larger vocabulary and diversity.

You can download it via torchtext or directly from HuggingFace datasets, or use the gensim.downloader API (e.g., api.load('text8')).

Preprocessing: The raw data requires cleaning. You must implement a preprocessing pipeline that removes punctuation, handles numbers (e.g., replace with), and lowers all cases.

1.2 Implementation details

We recommend that you train your word2vec models using the Gensim package. You are welcome to code up your own implementation (e.g. in PyTorch) if you like, but if you choose to do this, you are responsible for ensuring that your implementation is correct.

1.3 Bonus

For up to five points of extra credit, you may also do an additional evaluation and analysis. You can evaluate the above parameter settings on GloVe (in this case we recommend using the implementation of GloVe available from its website); Alternatively, you may implement a novel modification to word2vec or GloVe and evaluate it for up to five points of extra credit.

Answer to part 2-1

۱) کد Python در Google Colab قرار داده شد (لینک ضمیمه گردیده است)

- ✓ Click on link
- ✓ GitHub

ترجمه سوال «۱» به فارسی

⊙ تمرین۲-۱: ماجراجویی با تعبیه‌های واژه (Word Embeddings)

هدف این تمرین این است که درک عملی و نظریِ محکمی از ساز و کار داخلی تعبیه‌های واژه (**word embeddings**) به دست آورید در بخش اول، اثر ابر پارامترهای مختلف در الگوریتم‌های تعبیه‌واژه را بررسی می‌کنید در بخش دوم نیز از تعبیه‌های واژه در یک کار طبقه‌بندی استفاده خواهید کرد.

⊙ دستورالعمل‌های کلی

برای اجرای کامل آزمایش‌ها باید زمان کافی در نظر بگیرید؛ اجرای مجموعه کامل آزمایش‌ها ممکن است حداقل یک روز طول بکشد و در مرحله دیباگ نیز زمان اجرا افزایش پیدا کند توصیه می‌شود از چند روز قبل از موعد تحویل کدنویسی را شروع کنید.

۱) جست‌وجوی ابر پارامتر (Parameter Search)

ابر پارامترهایی که باید بررسی شوند:

- ❖ معماری **CBOW** و **Skip-gram**
- ❖ بُعد بردار **(Dimension)**: ۵۰، ۱۰۰، ۳۰۰
- ❖ اندازه پنجره **(Window size)**: ۵ و ۱۰
- ❖ **Epochs**: دقیقاً ۵ (همه مدل‌ها دقیقاً ۵ ایپاک آموزش ببینند)
- ❖ نکته: **Negative sampling** را ثابت و برابر ۵ نگه دارید.

این ترکیب‌ها در مجموع ۱۲ تنظیم یکتا ایجاد می‌کند (۲×۳×۲).

۲) ارزیابی (Evaluation)

اسکریپت ارزیابی آماده داده نمی‌شود و باید خودتان حلقه ارزیابی را با توابع داخلی **Gensim** پیاده‌سازی کنید:

- ❖ **WordSim353** با **evaluate_word_pairs()**

- **BATS** با **evaluate_word_analogies()**
- دانلود و بارگذاری درست دیتاست‌های **WordSim353** و **BATS** بر عهده شماست.

۳) گزارش (Writeup)

تحویل بخش اول شامل:

۱) یک جدول شامل نتایج جست‌وجوی پارامترها
۲) یک تحلیل نوشتاری از نتایج

در جدول هر ردیف مربوط به یک تنظیم است و ستون‌ها شامل معماری، اندازه پنجره، بُعد، تعداد **negative samples**، همبستگی **WordSim353**، دقت ۴ دسته منتخب **BATS**، و همبستگی روی **Win353paraphrase** است.

در تحلیل نوشتاری، حداقل باید به این پرسش‌ها پاسخ دهید:

۱) آیا بُعد بزرگ‌تر همیشه باعث عملکرد بهتر می‌شود؟ کجاها و چرا؟
۲) آیا بهتر شدن در یک تسک یعنی بهتر شدن در سایر تسک‌ها هم هست؟ چرا؟ و چرا نه؟
۳) آیا عملکرد در دسته‌های مختلف **analogy** مشابه است؟ اگر متفاوت است، چگونه و چرا؟ یک **error analysis** کوتاه انجام دهید.

۴) تحلیل کیفی (Qualitative)

- ۳ واژه چند معنایی (مثل **bank**، **apple**، **run**) انتخاب کنید
- ۵ همسایه نزدیک را در بهترین مدل و بدترین مدل پیدا کنید
- اثر **Window** را تحلیل کنید: پنجره کوچک بیشتر شباهت نحوی/صرفی می‌دهد یا معنایی؟ پنجره بزرگ چطور؟
- یافته‌ها را در قالب جدول گزارش کنید و کوتاه بحث کنید.

۵) داده آموزشی و پیش‌پردازش

به جای **Brown corpus**، مدل‌ها روی **WikiText-2** یا **text8** آموزش داده می‌شوند داده خام باید پاکسازی شود:

حذف علائم نگارشی، مدیریت عددها (مثل **<NUM>**)، و **lowercase** کردن.

هدف تمرین «۱» - Purpose of exercise 1

هدف بخش اول این است که نشان دهد انتخاب‌های اصلی در **Word2Vec** (معماری **CBOW/Skip-gram**، بُعد بردار، و اندازه پنجره) چگونه کیفیت تعبیه‌ها (**Embeddings**) را تغییر می‌دهند؛ هم در سنجش شباهت کلمه (**WordSim353** و **Win353-Para**) و هم در سنجش روابط قیاسی/آنالوژی (**BATS**)

در پایان نیز با تحلیل کیفی همسایه‌های نزدیک واژه‌های چندمعنایی، بررسی می‌کنیم مدل‌ها چه نوع اطلاعاتی را بهتر یاد گرفته‌اند (معنایی در برابر نحوی/صرفی)

گزارش نهایی «Part 1» - Report of the first exercise

۱) داده آموزشی، پیش‌پردازش و تنظیمات کلی

- **Training corpus:** `text8` (از طریق `gensim.downloader`)
- **Preprocessing pipeline:**
  - تبدیل همه توکن‌ها به حروف کوچک (**lowercasing**)
  - حذف علائم نگارشی (**punctuation removal**)
  - تبدیل عددها به توکن `<NUM>`

- تنظیمات ثابت آموزش:
  - `epochs = 5`
  - `negative = 5`
  - `min_count = 5` (طبق اجرای ما)

۲) فضای جست‌وجوی پارامترها

برای **Word2Vec**، ۱۲ مدل با ترکیب‌های زیر آموزش داده شد:

- **Architecture:** {CBOW، Skip-gram}
- **Dimension:** {۵۰, ۱۰۰, ۳۰۰}
- **Window size:** {۵, ۱۰}
- **Negative sampling:** ۵ (ثابت)
- **Epochs:** ۵ (ثابت)

۳) ارزیابی و معیارها

برای ارزیابی از توابع داخلی **Gensim** استفاده شد:

أ) WordSim353

- تابع: `model.wv.evaluate_word_pairs(wordsim353_path)`
- خروجی این تابع معمولاً شامل دو نوع همبستگی است (**Pearson** و **Spearman**)
- در این گزارش مطابق فایل خروجی ما، **Spearman** گزارش شده است (`WordSim353_Spearman`)

ب) Win353Paraphrase

- تابع: `model.wv.evaluate_word_pairs(win353_path)`
- مشابه **WordSim353**، خروجی شامل **Pearson** و **Spearman** است
- در این گزارش، **Spearman** گزارش شده است (`Win353_Spearman`)

ج) BATS (Analogy)

- تابع: `model.wv.evaluate_word_analogies(questions_file)`
- خروجی این تابع شامل **Accuracy** است و در جدول ما با ستون‌های `BATS_1...BATS_4` گزارش شده است
- نکته اجرایی: دیتاست **BATS** در اصل فایل‌های زوج‌واژه دارد، اما برای `evaluate_word_analogies` باید به فرمت سوال‌محور (**Google analogy style**) تبدیل شود در اجرای ما این تبدیل انجام شد و به همین دلیل ارزیابی **BATS** بدون خطا انجام شده است.

⊙ نکته مهم درباره OOV

- در خروجی ما:
  - `WordSim353_OOV% = 0.566572`
  - `Win353_OOV% = 0.566572`

یعنی حدود ۵۶/۷ درصد از زوج‌کلمات ارزیابی در واژگان مدل وجود نداشته‌اند این مقدار **OOV** بالا می‌تواند سقف **attainable** برای همبستگی‌ها را کاهش دهد و در تفسیر نتایج باید در نظر گرفته شود (به‌خصوص به علت `min_count=5` که بخشی از واژه‌های کم‌فراوانی را حذف می‌کند).

۴) جدول نتایج جست‌وجوی پارامترها (Table1)

- جدول زیر مستقیماً از **word2vec_param_search_results.csv** ما که در پایان تمرین به دست آوردیم استخراج شده است.
- توضیح **BATS**ها در این اجرا:
  - **BATS_1**: plural_reg
  - **BATS_2**: plural_irreg
  - **BATS_3**: derivational (noun+less_reg)
  - **BATS_4**: antonyms (binary)

ModelName	Architecture	Window	Dim	Neg	WordSim353 (Spearman)	WordSim OOV%	Win353-Para (Spearman)	Win353 OOV%	BATS_1	BATS_2	BATS_3	BATS_4	MeanScore
cbow_dim50_win5	cbow	5	50	5	0.5979	0.5666	0.4893	0.5666	0.5122	0.2856	0.0000	0.3137	0.4425
cbow_dim50_win10	cbow	10	50	5	0.6548	0.5666	0.5096	0.5666	0.5014	0.3046	0.0000	0.2484	0.4394
cbow_dim100_win5	cbow	5	100	5	0.6227	0.5666	0.5317	0.5666	0.6068	0.3840	0.0000	0.3725	0.4901
cbow_dim100_win10	cbow	10	100	5	0.6809	0.5666	0.5380	0.5666	0.6080	0.3682	0.0020	0.4510	0.5064
cbow_dim300_win5	cbow	5	300	5	0.6321	0.5666	0.5398	0.5666	0.6284	0.4286	0.0000	0.4052	0.5080
cbow_dim300_win10	cbow	10	300	5	0.6810	0.5666	0.5346	0.5666	0.6292	0.4024	0.0000	0.4314	0.5118
skipgram_dim50_win5	skipgram	5	50	5	0.6564	0.5666	0.5130	0.5666	0.5386	0.2782	0.0000	0.2484	0.4416
skipgram_dim50_win10	skipgram	10	50	5	0.6686	0.5666	0.5100	0.5666	0.4594	0.2652	0.0000	0.3072	0.4369
skipgram_dim100_win5	skipgram	5	100	5	0.6868	0.5666	0.5333	0.5666	0.5872	0.3700	0.0000	0.4967	0.5107
skipgram_dim100_win10	skipgram	10	100	5	0.6997	0.5666	0.5333	0.5666	0.5524	0.3294	0.0000	0.4118	0.4842
skipgram_dim300_win5	skipgram	5	300	5	0.6829	0.5666	0.5476	0.5666	0.5888	0.4044	0.0139	0.5359	0.5264
skipgram_dim300_win10	skipgram	10	300	5	0.7109	0.5666	0.5396	0.5666	0.5916	0.3758	0.0059	0.5163	0.5192

Table 1 - Results of parameter search (12 settings)

۵) انتخاب بهترین و بدترین مدل

با معیار **MeanScore**:

- بهترین مدل: **skipgram_dim300_win5** با **MeanScore = 0.5264**
- بدترین مدل: **skipgram_dim50_win10** با **MeanScore = 0.4369**

۶) تحلیل نتایج (پاسخ مستقیم به سوال‌ها)

۱) آیا بُعد بزرگ‌تر همیشه بهتر است؟

در نتایج مشاهده می‌شود که افزایش **Dim** از ۵۰ به ۱۰۰ و سپس ۳۰۰ به‌طور کلی باعث بهبود در **WordSim/Win353** و برخی دسته‌های **BATS** شده است با این حال میزان بهبود از ۱۰۰ به ۳۰۰ معمولاً کمتر از جهش ۵۰ به ۱۰۰ است (بازده نزولی)

توضـیح: بعد بالاتر ظرفیت نمایش روابط بیشـتری را می‌دهد، اما با داده و **epochs** ثابت، بهبود می‌تواند محدود شـود و حتی برخی روابط (مثل بعضی **derivation**ها) همچنان سخت باقی بمانند.

۲) آیا بهتر شدن در یک تسک یعنی بهتر شدن در سایر تسک‌ها؟

خیر لزوماً نه چون ماهیت تسک‌ها متفاوت است:

- **WordSim/Win353** بیشتر شباهت معنایی/توزیعی را می‌سنجد.
- **BATS** روابط قیاسی را می‌سنجد که در برخی دسته‌ها (مورفولوژی/اشتقاق) به ساختارهای قاعده‌مند متکی است.
- بنابراین ممکن است مدلی در **WordSim** بهتر باشد ولی در یک دسته خاص **BATS** عملکرد ضعیف‌تری داشته باشد (و بالعکس).

۳) آیا عملکرد در دسته‌های BATS مشابه بود؟ (به همراه error analysis کوتاه)

عملکرد در دسته‌ها متفاوت بود:

- **plural_reg (BATS_1)** معمولاً بهتر از **plural_irreg (BATS_2)** بود، چون جمع منظم قاعده‌مندتر است و مدل راحت‌تر یاد می‌گیرد جمع نامنظم استثناهای بیشتری دارد.
- **antonyms (BATS_4)** نسبتاً بهتر از دسته **derivational** بوده، اما همچنان سخت است چون متضادها در متن اغلب در زمینه‌های مشابه ظاهر می‌شوند و مدل‌های توزیعی ممکن است آن‌ها را نزدیک ببینند (**distributional similarity**).
- **BATS_3 (noun+less_reg)** تقریباً در اکثر تنظیمات نزدیک صفر مانده است این می‌تواند ناشی از چند عامل باشد:

۱. بسیاری از کلمات این رابطه ممکن است **OOV** باشند یا کم‌فراوانی و حذف شده باشند (**min_count=5**)
۲. رابطه اشتقاقی لزوماً یک **offset** خطیِ پایدار برای همه مثال‌ها ندارد
۳. حساسیت این دسته به واژگان و پوشش داده می‌تواند بسیار بالا باشد

۷) تحلیل کیفی Polysemy و اثر Window --(Table 2)

جدول زیر از `polysemous_neighbors.csv` ما استخراج شده است (۵ همسایه نزدیک در بهترین و بدترین مدل)

Word	Best model neighbors (best=skipgram_dim300_win5)	Worst model neighbors (worst=skipgram_dim50_win10)
bank	monetary, banks, fund, suntrust, banking	banks, monetary, fund, banking, loans
apple	macintosh, iic, iigs, iie, amiga	macintosh, imac, iigs, iic, workstation
run	runs, running, ran, cooperatively, consecutively	running, drivers, lotteries, runs, backups

Table 2 - Nearest neighbors for polysemous words (best vs worst)

⊙ بحث (اثر window)

با توجه به اینکه بهترین مدل `window=5` و بدترین مدل `window=10` دارد، در نمونه‌ی **run** دیده می‌شود که پنجره کوچک‌تر همسایه‌های نزدیک‌تری از نظر صرفی/نحوی (**runs/running/ran**) تولید کرده، اما پنجره بزرگ‌تر برخی همسایه‌های زمینه‌ای/موضوعی و گاهی نامرتبط (**drivers/lotteries/backups**) را وارد کرده است این رفتار با شهود رایج سازگار است: پنجره کوچک‌تر تمایل بیشتری به الگوهای محلی (نحوی/صرفی) دارد و پنجره بزرگ‌تر می‌تواند زمینه‌های معنایی گسترده‌تری را وارد کند، که گاهی باعث مخلوط شدن کاربردهای مختلف واژه می‌شود.

۸) جمع‌بندی نهایی

در پارت اول تمرین ۱۲ مدل **Word2Vec** با ترکیب معماری، بُعد و اندازه پنجره آموزش داده شد نتایج نشان داد افزایش بُعد عموماً عملکرد را بهتر می‌کند و بهترین تنظیم ما **Skip-gram** با `dim=300` و `window=5` بوده است همچنین مشاهده شد که عملکرد بین تسک‌ها کاملاً همسو نیست و دسته‌های **BATS** سختی‌های متفاوتی دارند (به‌خصوص دسته **derivational**) تحلیل کیفی نیز نشان داد اندازه پنجره می‌تواند بر نوع همسایه‌ها (محلی/صرفی در برابر زمینه‌ای/موضوعی) اثر بگذارد.

Homework 2-2: Downstream Task Evaluation

One of the most popular usages of word embeddings is in classification tasks. Instead of relying on existing repositories, you will build a classifier from scratch to categorize text topics.

The Dataset: We have provided a dataset named AG_News_Subset.csv (or you can download the "AG News" dataset from Kaggle/Torchtext). This dataset contains news articles classified into 4 categories: World, Sports, Business, and Sci/Tech.

Your Task: You need to compare three different feature extraction methods for a Logistic Regression (or SVM) classifier:

1. **Baseline:** TF-IDF Vectorization (Bag-of-Words approach).
2. **Your Best Model:** Use the best performing Word2Vec model you trained in Part 1. Represent each document by **averaging** the word vectors of its tokens.
3. **Pretrained GloVe:** Use glove-wiki-gigaword-100 (available via Gensim). Represent documents by averaging the vectors.

Requirements:

- For the embedding-based methods, you must handle "Out-of-Vocabulary" (OOV) words appropriately (explain your strategy in the report).
- Report **Accuracy, Precision, Recall, and Macro-F1 Score** for all three methods in a comparison table.
- **Analysis:** Why does TF-IDF perform better or worse than averaged word embeddings in this specific task? Does the "World" category get confused with "Business" more often in one model than another? (Show a Confusion Matrix)

Reza Jafari (MSc Software student)

Department of Computer Engineering

Answer to part 2-2

۱) کد Python در GoogleColab قرار داده شد (لینک ضمیمه گردیده است)

- ✓ Click on link
- ✓ GitHub

ترجمه سوال «۲» به فارسی

⊙ **تمرین ۲-۲: ارزیابی در کار پایین دستی**

یکی از رایج‌ترین کاربردهای تعبیه‌های واژه **(Word Embeddings)** استفاده از آن‌ها در کارهای طبقه‌بندی **(Classification)** است در این تمرین به‌جای اتکا به مخازن آماده باید یک طبقه‌بند را از ابتدا بسازید تا متن‌ها را بر اساس موضوع دسته‌بندی کند.

⊙ **دیتاست**

دیتاستی با نام **AG_News_Subset.csv** در اختیار شما قرار داده شده است (یا می‌توانید دیتاست «**AG News**» را از **Kaggle/Torchtext** دریافت کنید) این دیتاست شامل خبرهایی است که در ۴ دسته طبقه‌بندی شده‌اند: **World**، **Sports**، **Business** و **Sci/Tech**

⊙ **وظیفه شما**

باید سه روش مختلف استخراج ویژگی را برای یک طبقه‌بند **LogisticRegression** (یا **SVM**) مقایسه کنید:

۱. **روش پایه: TF-IDF** (رویکرد **Bag-of-Words**)
۲. **بهترین مدل شما:** استفاده از بهترین **Word2Vec** آموزش‌داده‌شده در تمرین ۱ و نمایش هر سند با میانگین بردارهای کلمات.
۳. **مدل از پیش‌آموزش‌دیده:** استفاده از **glove-wiki-gigaword-100** و نمایش هر سند با میانگین بردارها.

⊙ **الزامات**

باید برای روش‌های مبتنی بر **embedding** مسئله‌ی **OOV** را مدیریت کرده و راهبردتان را توضیح دهید معیارهای **Accuracy**، **Precision**، **Recall** و **Macro-F1** را برای هر سه روش در جدول مقایسه گزارش کنید، و با تحلیل **Confusion Matrix** توضیح دهید چرا **TF-IDF** بهتر/بدتر از میانگین **embedding**ها عمل می‌کند و آیا کلاس **World** در یک مدل بیشتر با **Business** اشتباه می‌شود یا خیر.

هدف تمرین «2» - Purpose of exercise 2

هدف این تمرین بررسی اثر «نوع نمایش متن» بر عملکرد طبقه‌بندی موضوع اخبار است در اینجا یک مدل طبقه‌بندی ثابت (**LogisticRegression**) نگه داشته می‌شود و فقط نمایش متن تغییر می‌کند تا مشخص شود کدام نمایش برای این مسئله مناسب‌تر است علاوه بر مقایسه عددی معیارها، تحلیل **Confusion Matrix** کمک می‌کند کیفیت مدل تنها به یک عدد خلاصه نشود و به صورت دقیق‌تر روشن شود که خطاها بین کدام کلاس‌ها رخ می‌دهد به‌خصوص بررسی می‌شود آیا **World** بیشتر با **Business** اشتباه می‌شود یا خیر همچنین نقش **OOV** در روش‌های **embedding-based** بررسی می‌شود چون پوشش واژگانی پایین می‌تواند باعث شود بردار سند اطلاعات کافی نداشته باشد و دقت کاهش پیدا کند.

گزارش نهایی «Part 2» - Report of the second exercise

۱) داده و پیش‌پردازش

دیتاست **AG_News_Subset.csv** شامل ۷۶۰۰ نمونه خبر است متن نهایی هر خبر با اتصال **Title** و **Description** ساخته شد تا اطلاعات موضوعی کامل‌تری وارد مدل شود توزیع کلاس‌ها متوازن است و هر کلاس ۱۹۰۰ نمونه دارد داده‌ها با نسبت ۸۰/۲۰ به آموزش و آزمون تقسیم شدند و برای حفظ توازن کلاس‌ها از **stratify** استفاده شد. بنابراین مجموعه آزمون ۱۵۲۰ نمونه دارد و از هر کلاس دقیقاً ۳۸۰ نمونه در تست وجود دارد.

۲) مدل طبقه‌بندی

در هر سه روش از **Logistic Regression** استفاده شد تا مقایسه به شکل منصفانه انجام شود و تفاوت عملکرد صرفاً به تفاوت نمایش متن نسبت داده شود این طراحی آزمایش اجازه می‌دهد اثر ویژگی‌ها به‌صورت واضح دیده شود.

۳) روش‌های استخراج ویژگی

سه نوع ویژگی استخراج شد:

- **TF-IDF** به عنوان نمایش **Bag-of-Words**
- میانگین بردارهای سند با بهترین **Word2Vec** تمرین ۱
- میانگین بردارهای سند با **GloVe-100** از پیش‌آموزش‌دیده

در روش‌های **embedding-based** نمایش هر سند با میانگین‌گیری از بردار توکن‌های داخل سند ساخته شد.

۴) مدیریت OOV

در روش‌های **Word2Vec(avg)** و **GloVe(avg)**، توکن‌های خارج از واژگان (**OOV**) حذف شـدند و فقط توکن‌های موجود در واژگان مدل در میانگین شرکت داده شدند اگر یک سند هیچ توکن **in-vocab** نداشت، بردار آن سند برابر بردار صفر در نظر گرفته شـد میانگین **OOV** در تسـت برای **Word2Vec** حدود ۰.۰۶۱۱ و برای **GloVe** حدود ۰.۰۳۶۰ بود این اختلاف نشـان می‌دهد **GloVe** پوشش واژگانی بیشتری دارد و همین موضوع می‌تواند نمایش سند را پایدارتر کند.

۵) نتایج کمی

معیارهای **Accuracy** و نیز **Precision/Recall/F1** به صورت **Macro-average** گزارش شد تا هر کلاس وزن برابر داشته باشد نتایج نشان داد **GloVe(avg)** اندکی بهتر از **TF-IDF** و هر دو بهتر از **Word2Vec(avg)** عمل کرده‌اند.

Method	Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Avg-OOV-Test
TF-IDF + LR	0.882237	0.881546	0.882237	0.881537	-----
Best Word2Vec(avg) + LR	0.871711	0.871056	0.871711	0.871321	0.061080
GloVe-100(avg) + LR	0.884868	0.884510	0.884868	0.884621	0.035978

Table 1 - Comparison of Three Methods (Logistic Regression Classifier)

۶) تحلیل Confusion Matrix (Figures 1–3)

در **Confusion Matrix** سطرها کلاس واقعی و ستون‌ها کلاس پیش‌بینی‌شده هستند بنابراین هر مقدار خارج از قطر اصلی نشان‌دهنده یک نوع اشتباه است چون در مجموعه آزمون برای هر کلاس دقیقاً ۳۸۰ نمونه داریم، اعداد خطا را می‌توان مستقیماً به درصد نیز تفسیر کرد و شدت اشتباه‌ها را بهتر فهمید.

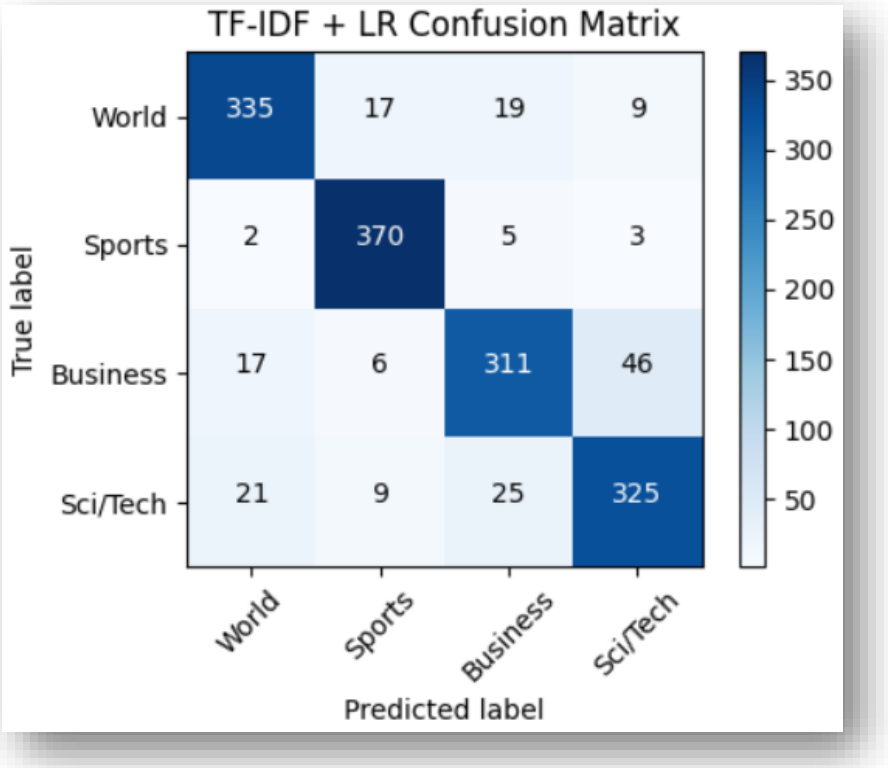

Figure 1 - TF-IDF + Logistic Regression

شکل ۱ (Figure 1) — TF-IDF + Logistic Regression

در **TF-IDF** الگوی خطا نشان می‌دهد بیشترین ابهام بین **Business** و **Sci/Tech** رخ می‌دهد به طور مشخص، **Business** → **Sci/Tech = 46** یعنی حدود ۱۲.۱% از خبرهای **Business** به اشتباه **Sci/Tech** تشخیص داده شده‌اند و **Sci/Tech** → **Business = 25** یعنی حدود ۶.۶% از **Sci/Tech** به **Business** تبدیل شده است این عدم تقارن می‌تواند ناشی از این باشد که بسیاری از متن‌های **Business** شامل واژه‌ها و مفاهیمی هستند که در **Sci/Tech** هم زیاد دیده می‌شوند (مثلاً شرکت‌ها، محصولات، فناوری و بازار) درباره هدف خاص تمرین میزان اشتباه **World** و **Business** در این روش نسبتاً کنترل‌شده است؛ **World** → **Business = 19** و **Business** → **World = 17** که هر دو نزدیک به ۵% هستند بنابراین **TF-IDF** بیشتر در مرز **Business/SciTech** آسیب می‌بیند تا در مرز **World/Business**.

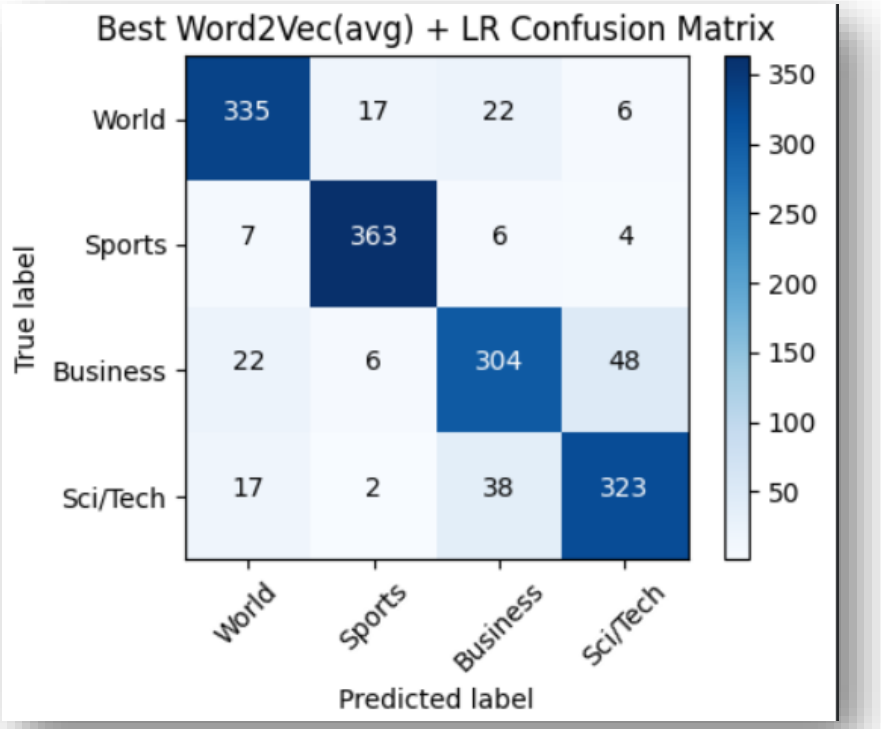

Figure 2 - Best Word2Vec(avg) + Logistic Regression

- شکل ۲ **(Figure 2) — Best Word2Vec(avg) + Logistic Regression**

در **Word2Vec(avg)** اختلاف **Business** و **Sci/Tech** پررنگ‌تر می‌شود و این دقیقاً با ماهیت «میانگین‌گیری **embedding**ها» سازگار است چون میانگین‌گیری معمولاً وزن ویژه‌ای برای واژه‌های کلیدی بسیار تمایزبخش مثل **TF-IDF** قائل نیست و حضور واژه‌های عمومی‌تر می‌تواند اثر کلیدواژه‌ها را رقیق کند در این روش مقدار **Sci/Tech : 48** → **Business** و **Sci/Tech** → **Business : 38** است؛ یعنی در هر دو جهت خطا قابل توجه است و به‌خصوص جهت **Sci/Tech**→**Business** نسبت به **TF-IDF** افزایش پیدا کرده است علاوه بر آن اشتباه بین **World** و **Business** نیز کمی بیشتر دیده می‌شود (**World** → **Business = 22** و **Business** → **World = 22**) یکی از تفسیرهای منطقی این وضعیت هم‌پوشانی معنایی زیاد بین Business و **Sci/Tech** در فضای برداری و همچنین اثر **OOV** بالاتر **Word2Vec** در مقایسه با **GloVe** است که می‌تواند کیفیت بردار سند را کاهش دهد.

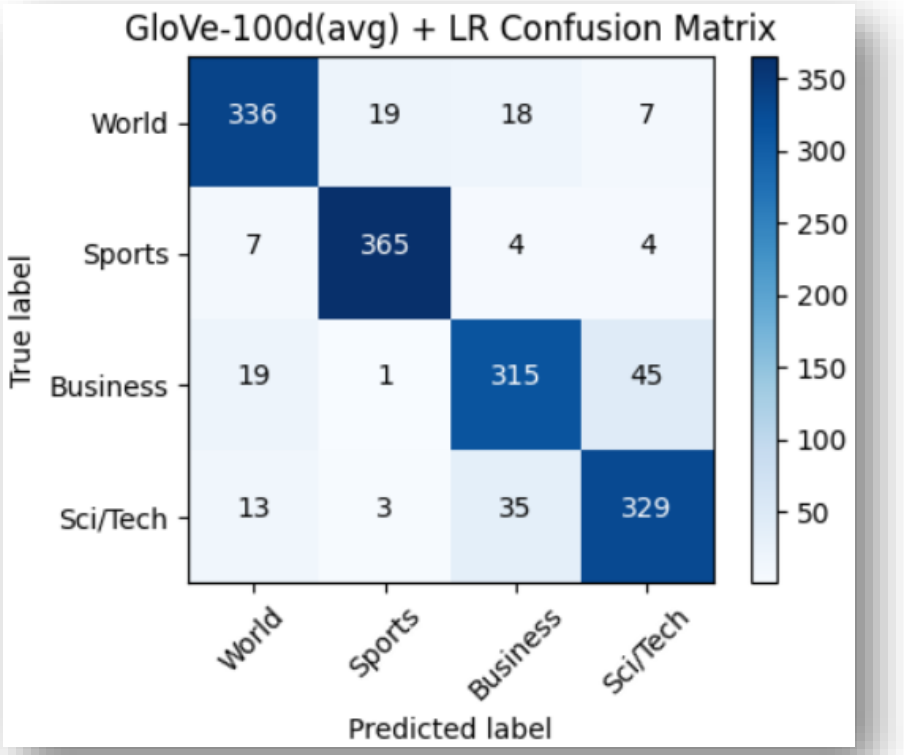

Figure 3 - GloVe-100(avg) + Logistic Regression

- شکل ۳ **(Figure 3) — GloVe-100(avg) + Logistic Regression**

در **GloVe(avg)** بهترین عملکرد کلی دیده می‌شود و بخشی از خطاهای **Word2Vec** کاهش یافته است هرچند بزرگ‌ترین منبع خطا همچنان **Business/SciTech** است اما مقدارها کمی کمتر و پایدارتر هستند: **Business → Sci/Tech = 45** و **Sci/Tech → Business = 35** این نتیجه با **OOV** کمتر **GloVe** نیز هم‌خوان است چون وقتی واژه‌های بیشتری پوشش داده شوند میانگین بردارها نماینده بهتری از متن می‌شود و طبقه‌بند تصمیم دقیق‌تری می‌گیرد برای **World/Business** نیز اعداد نزدیک به **TF-IDF** باقی می‌ماند (**World → Business = 18** و **Business → World = 19**) که نشان می‌دهد **GloVe(avg)** در این مرز حساس عملکردی تقریباً هم‌سطح **TF-IDF** دارد و از **Word2Vec(avg)** بهتر است.

در مجموع **Confusion Matrix**ها نشان می‌دهند که در هر سه روش مرز **Business** و **Sci/Tech** اصلی‌ترین نقطه ابهام است و مرز **World** و **Business** خطای ثانویه محسوب می‌شود تفاوت مهم این است که **TF-IDF** معمولاً خطاهای **Business→Sci/Tech** کمتری دارد، **Word2Vec(avg)** این خطاها را افزایش می‌دهد و **GloVe(avg)** با پوشش واژگانی بهتر و نمایش پایدارتر، بخشی از این افزایش را جبران می‌کند.

۷) تحلیل نهایی: چرا TF-IDF بهتر/بدتر از میانگین embeddingهاست؟

TF-IDF معمولاً برای طبقه‌بندی موضوعی اخبار قوی است چون به کلمات کلیدی هر حوزه وزن بالا می‌دهد و واژه‌های عمومی را کم‌اثر می‌کند در مقابل میانگین **embedding**ها ترتیب و ساختار جمله را حذف می‌کند و همچنین سیگنال کلیدواژه‌ها را در میانگین رقیق می‌سازد بنابراین در کلاس‌هایی با هم‌پوشانی معنایی (مانند **Business** و **Sci/Tech**) احتمال اشتباه بالاتر می‌رود با این حال **GloVe(avg)** به دلیل آموزش روی داده بزرگ و پوشش واژگانی بهتر (**OOV** کمتر) توانست در این اجرا کمی بهتر از **TF-IDF** ظاهر شود و نشان دهد حتی یک نمایش ساده میانگین اگر **embedding** با کیفیت و پوشش مناسب داشته باشد می‌تواند رقابتی باشد.

۸) جمع‌بندی

در این تمرین سه روش نمایش متن برای طبقه‌بندی اخبار با **Logistic Regression** مقایسه شد **GloVe-100(avg)** بهترین نتیجه را به دست آورد و **TF-IDF** بسیار نزدیک به آن بود در حالی که **Word2Vec(avg)** ضعیف‌تر عمل کرد تحلیل **Confusion Matrix** نشان داد بزرگ‌ترین ابهام در هر سه روش بین **Business** و **Sci/Tech** رخ می‌دهد و اشتباه **World↔Business** کمتر اما قابل مشاهده است مدیریت **OOV** مطابق الزام تمرین انجام شد (حذف توکن‌های **OOV** و بردار صفر برای اسناد کاملاً **OOV**) و تفاوت **OOV** بین **Word2Vec** و **GloVe** یکی از عوامل مهم در تفسیر تفاوت نتایج بود.