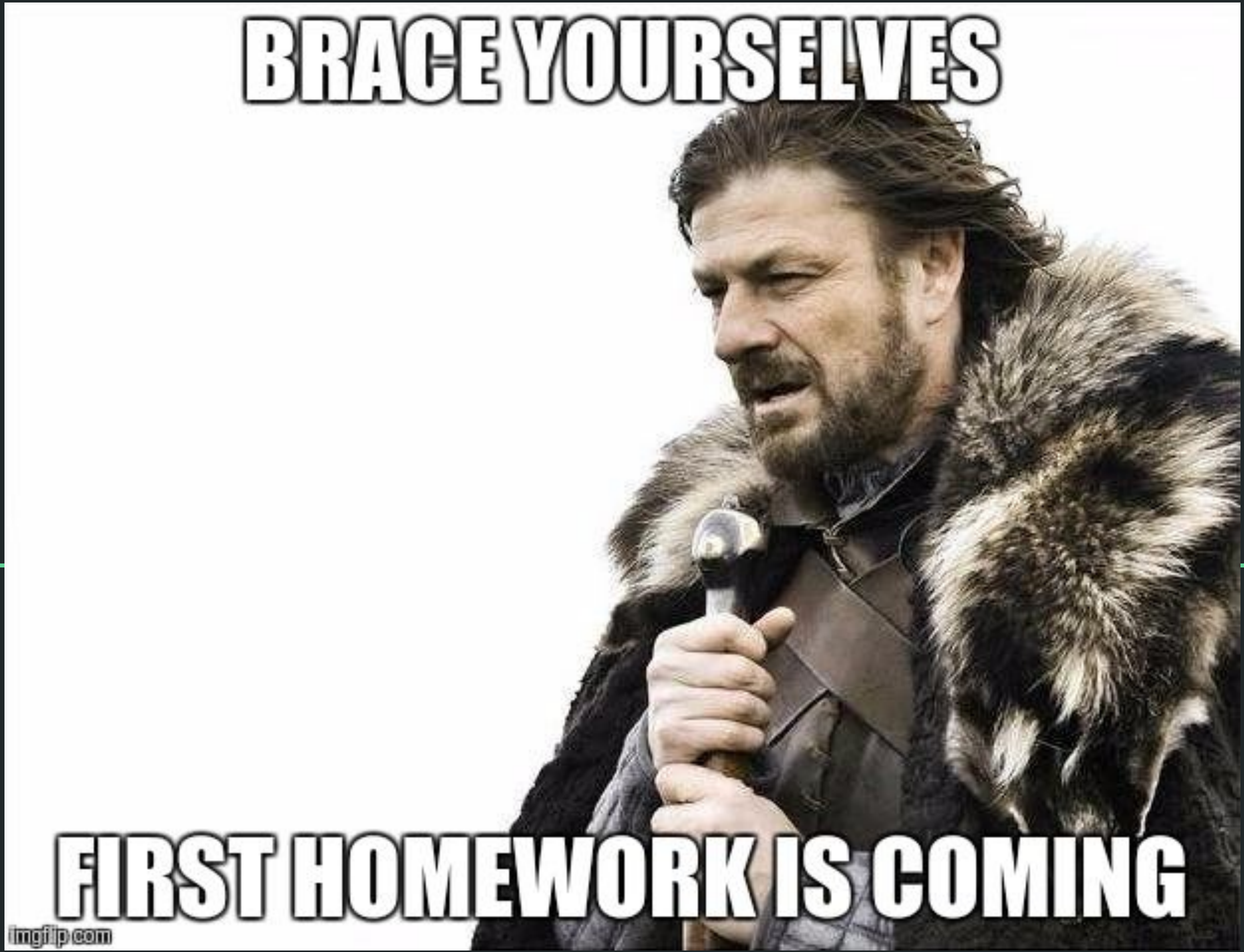

BRACE YOURSELVES
FIRST HOMEWORK IS COMING
imgflip.com

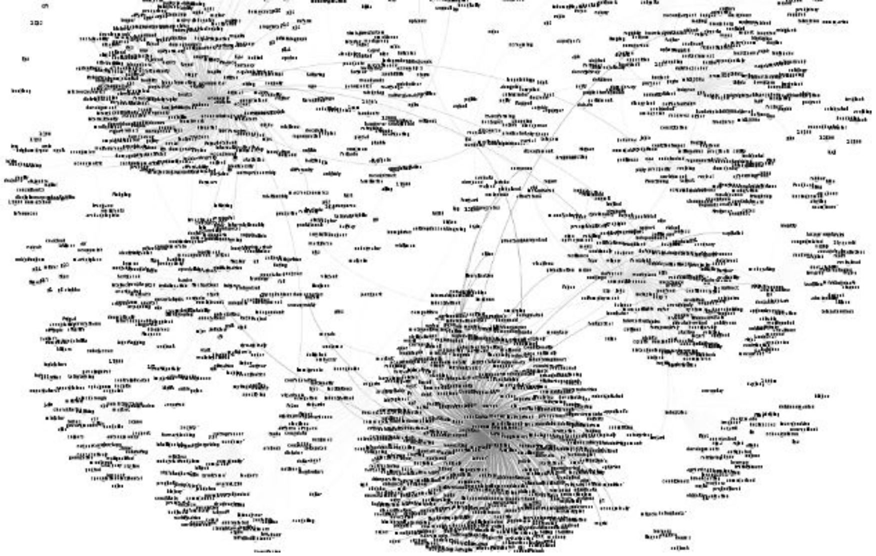
Homework 1

Federico Scozzafava & Valentina Pyatkin

(scozzafava | pyatkin)@di.uniroma1.it

What you will do:

- Create your own word embeddings
- Participate in a competition with prizes!



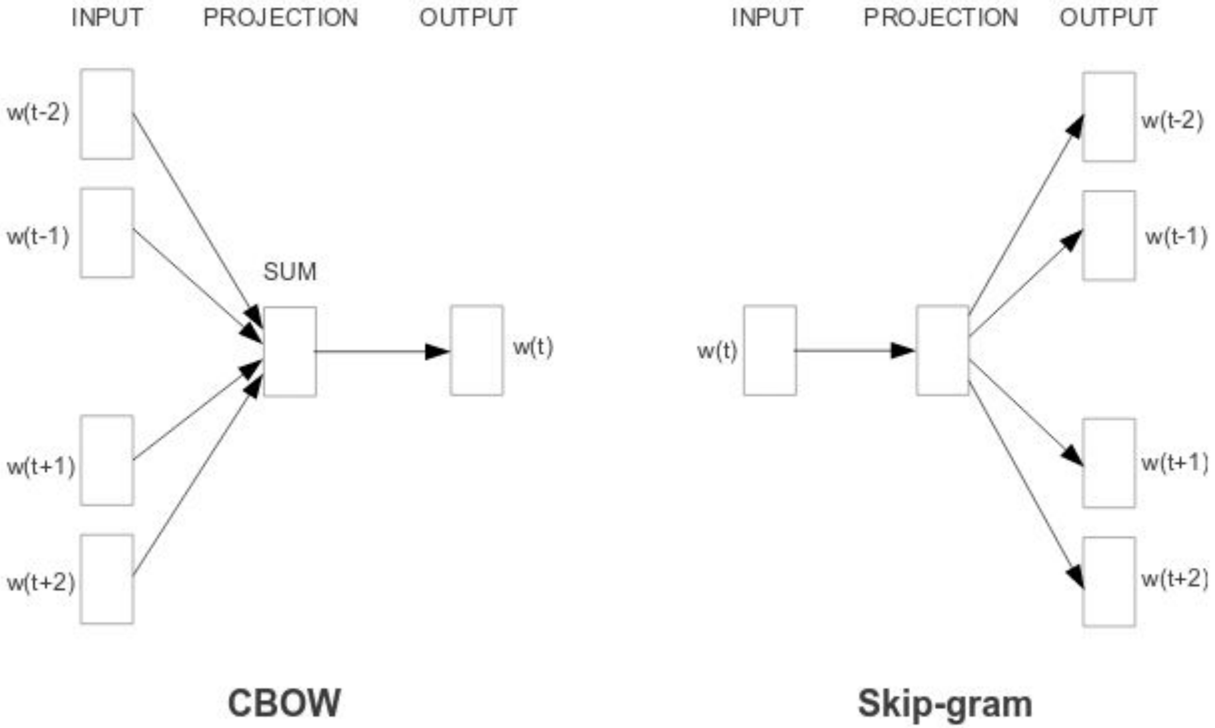
Structure of the Homework

- Implement a **Word2Vec model in Tensorflow**
  - we provide you with a skeleton and you have to fill in the functions
- **Train** it using the appropriate parameters you obtained through **grid-search**
- Answer a set of questions about your work in a 1 (max.!) page **report**
  - we will give you a list of questions you should answer
  - we will stop reading after the first page
- Use your vectors to solve the **domain-identification task**
- **Submit** your code, report, your vector visualization and your answers to the domain-identification task

What we provide:

- train, dev, test data (without labels obviously) for the domain-identification task:
- a development function to optimize your word-vectors, which prints the accuracy
- skeleton-code where you have to fill in specific functions (see comments in code and next slides)
- these slides as guidelines

Skip-gram vs. CBOW (Mikolov et al. 2013)



- Skip-gram
  - from target to context
  - better for larger datasets
- CBOW
  - from context to target
  - better for smaller datasets: "smoothes over a lot of the distributional information"

Parameters and Hyper-parameters

These are the parameters you will have to optimize using grid-search :
window-size, iteration number, embedding size, batch-size, number of negative samples (skipgram), vocabulary size

- **Grid-Search:** Train your model multiple times, changing one parameter at a time and evaluating each parameter configuration, in order to find the best parameters.
    - You will evaluate your vectors on a word similarity task for this purpose.
    - You will never be able to try out all configurations (there is simply not enough time for this), so choose 2-3 options for each parameter, e.g. try 150, 280, 400 for embedding size

Parameters and Hyper-parameters

These are the parameters you will have to optimize using grid-search :
window-size, iteration number, embedding size, batch-size, number of negative samples (skipgram), vocabulary size

- window-size: What is the context you are trying to predict (skip-gram) or you take as input (CBOW)?
- embedding-size: What is the size of your hidden layer?
- batch-size: How many examples does your model see before it updates its weights?
- iteration: How many times do you show your model the same example?
- number of negative samples: How many negative examples do you show your model? (only if you implement skip-gram)
- vocab. size: how many of the most-common words do you consider?

Questions for the report, Part 1: W2V

1. Shortly describe which W2V model you implemented (skip-gram or cbow) and explain the architecture of your model.
2. Fine-tune the following parameters: **window-size, iteration number, embedding size, batch-size, number of negative samples (if skipgram), vocabulary size**
    a. Plot the model's accuracy and loss while changing the models parameters
    b. Report your general findings in the report and report the most optimal parameter settings you found.

Explaining the functions: DataPreprocessing

- build_dataset:
    - Generates the dataset and dictionary data structures from the training set
    - The dictionary maps each word to an unique integer representation (index)
    - The dataset is the original training set where the words are replaced with their codes in dict
    - Handle the unseen words case by defining a special word ‘UNK’ in the dictionary
        - Assign UNK to the less frequent words in the dictionary
- generate_batch
    - For each training step generate the training samples and labels for the current batch
    - For each target word generate the pair <target,label> for each context word in the window
    - You can choose to implement **skipgram** or **cbow**

Explaining the functions: W2V

- Define the model structure in Tensorflow
  - Create the input and label placeholder
  - Embedding layer size
  - Weights and biases
- Define the initialization values for each layer
- Specify the loss function
  - Include the **noise-contrastive estimation** (only if you implement skip-gram)
- Choose the optimizer and learning rate

Evaluating the model

- We provide a function that prints some statistics about the embedding during the training
- The accuracy is measured performing the Analogical Reasoning task:
    - Example: `king is to queen as father is to ?`
- This is a really tough task! the accuracy depends mostly on the quality of the input data (e.g. tokenization), parameters and training time.
- Look at the examples and accuracy while tuning the parameters to verify that your model is training correctly, also check if your loss is decreasing.

Questions for the report, Part 2: Using your Vectors

1. Train your model using your final parameter settings and visualize your vectors on tensorboard. Include a nice/interesting picture of the visualization and write 2-3 sentences about what you found interesting in your word vector space.
2. Include the 5 most similar words for each of these words: "german", "most", "general", "food", "cat", "eat", "teach". What do you notice? Do they make sense?

Domains

- We classified Wikipedia pages into 34 domains (ANIMALS, MEDIA, SPORT_AND_RECREATION, HISTORY etc.)
- Each Wikipedia page is assigned to one domain
- We will give you the data in the following format:
  - 3 Folders: TRAIN, DEV and TEST (70%, 15%, 15%)
  - Each of these folders contains 34 subfolders, one for each domain
  - Each domain folder contains files representing wikipedia pages, the file-id is shown in the title and the file contains text from a wikipedia page
  - The TEST folder is structured differently: it contains all the files you will have to classify

Ideas/Hints on how to classify the domain

- Centroid-based approach:
    - represent your document as a vector by performing vector operations with your word-vectors, e.g. try adding them, averaging them, finding the centroid etc.
    - Compute the similarity between the domain and document vectors
    - calculate kNN
- ML approach:
    - choose a machine learning algorithm of your liking (SVM, log-regression, Random Forest) and use your word-vectors as features

Questions for the report, Part 3: Using your Vectors as Features for Domain Classification

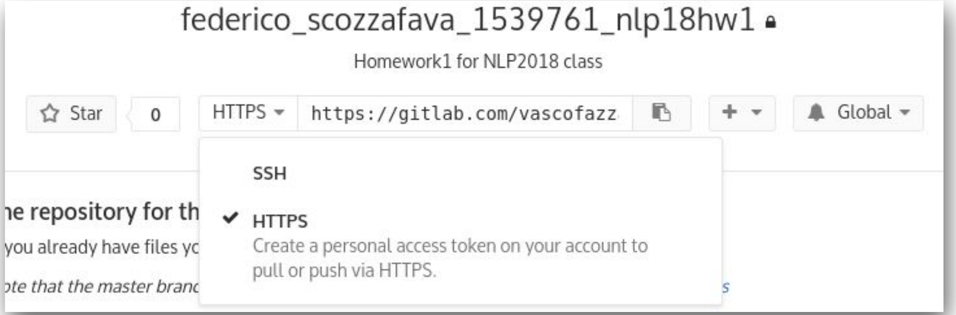
1. How did you use your vectors to classify the wikipedia domains? Describe your process.
2. How well does your domain classification system work? Mention the Recall, Precision and F1-measure on the development set, for each domain. Create a confusion matrix with your results.

What you have to submit:

- A zipped folder containing:
  - your report (1 page) as a pdf: report.pdf
  - an image of your vector visualization: vecviz.png
  - your source code as a GitLab repository (shared with us)
  - your answers for the domain-identification task on the test data: test_answers.tsv
    - in the following **format**: **id\tanswer\n** (id **TAB** answer **NEWLINE**) ('id' is the number in the filename, 'answer' is the domain name)
      - for example:
        - 86876 ANIMALS
        - 101374 CULTURE_AND_SOCIETY
    - if your answer file is in the wrong format it cannot be considered for the competition

Source code submission

- Register to GitLab.com and create a new project
- Name the project **firstname_lastname_matricola_nlp18hw1**
- Share the project with (project setting -> members):
  - pyatkin@di.uniroma1.it
  - federico.scozzafava@gmail.com
  - navigli@di.uniroma1.it
- Upload your code on the repository
- Get the **HTTP URL**



federico_scozzafava_1539761_nlp18hw1

Homework1 for NLP2018 class

☆ Star 0 HTTPS ▾ https://gitlab.com/vascofazz

SSH

✓ HTTPS

Create a personal access token on your account to pull or push via HTTPS.

ne repository for th

you already have files yo

ote that the master branc

How we will grade:

- The maximum grade for this homework is 34.5 (115% of 30) weighted as follows:
    - Quality, comments and cleanness of code [**40%**]
    - Report [**60%**]
    - Overall performance of the system [**15%**] (this really depends on how much time you spend tuning parameters)
- The competition scores will be published after the homework evaluations
- The top 5 ranked students will receive a (super!) fancy **BabelNet** T-Shirt!

We will check all your submissions for **plagiarism** with a plagiarism software!

- If we found that you plagiarised: you are **OUT** of this year's course and you cannot take the exam, you will have to sign up for the course next year.
- We have a zero-tolerance policy for plagiarism!
    - we found a couple of students who plagiarized last year and found it absolutely unacceptable

Timeline

- **Advice:**
  - start as early as possible training the vectors!
- Training a Neural Network takes time, especially if you train on CPU
- Memory is also an issue: you will probably not be able to load the whole training data into memory, e.g. you will have to create batches and load them separately.
- **Suggestion:** Start implementing the W2V asap and train it until you are satisfied with your vectors. The rest of the task should take less time.

Deadlines

- We will upload everything you need to complete this homework Sunday evening the latest, on the facebook group
- Your deadline for homework 1 will be: 15.04.18, 24:00
- The link where you have to submit:

Submission Form

Good Luck!

If you have any questions, do not hesitate to post them on the facebook group.