

NLP course 2021

Homework 1

Word-in-Context Disambiguation

Prof. Roberto Navigli

Teaching assistants:

- Cesare Campagnano
- Pere-Lluís Huguet Cabot



SAPIENZA
UNIVERSITÀ DI ROMA

**SAPIENZA
NLP**



Word-in-Context Disambiguation

A gentle introduction



Word-in-Context Disambiguation (WiC)

WiC is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses.

Word-in-Context Disambiguation (WiC)

WiC is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses.

- Use the **mouse** to click on the button

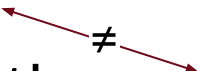
Word-in-Context Disambiguation (WiC)

WiC is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses.

- Use the **mouse** to click on the button
- The cat eats the **mouse**

Word-in-Context Disambiguation (WiC)

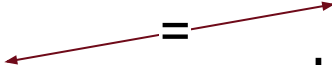
WiC is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses.

- Use the **mouse** to click on the button
 - The cat eats the **mouse**
- 

In the above examples, the word **mouse** has different meanings, because it is used in different contexts.

Word-in-Context Disambiguation (WiC)

WiC is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses.

- The cat eats the **mouse**
 - the **mouse** escaped from the predator
- 

In the above examples, the word **mouse** has different meanings, because it is used in different contexts.

WiC: context sentences, target words

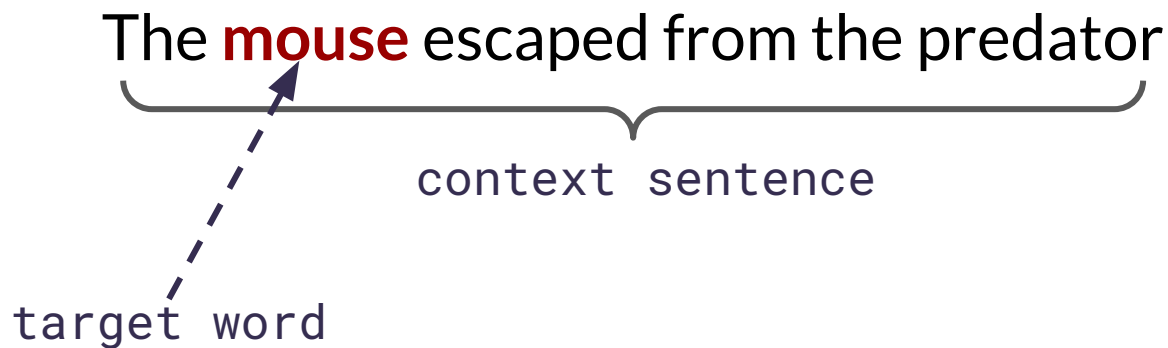
The mouse escaped from the predator

WiC: context sentences, target words

The mouse escaped from the predator

context sentence

WiC: context sentences, target words



The homework

Word-in-Context Disambiguation



Homework 1 - Word-in-Context Disambiguation



The goal

The task is formulated as a **binary classification task**:

Given two context sentences, determine whether the indicated target words have the same meaning.

The goal

The task is formulated as a **binary classification task**:

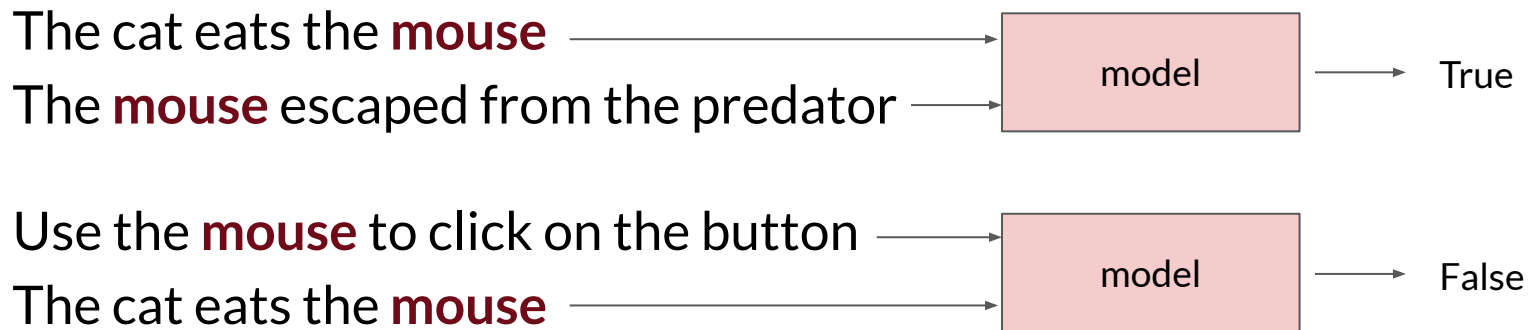
Given two context sentences, determine whether the indicated target words have the same meaning.



The goal

The task is formulated as a **binary classification task**:

Given two context sentences, determine whether the indicated target words have the same meaning.

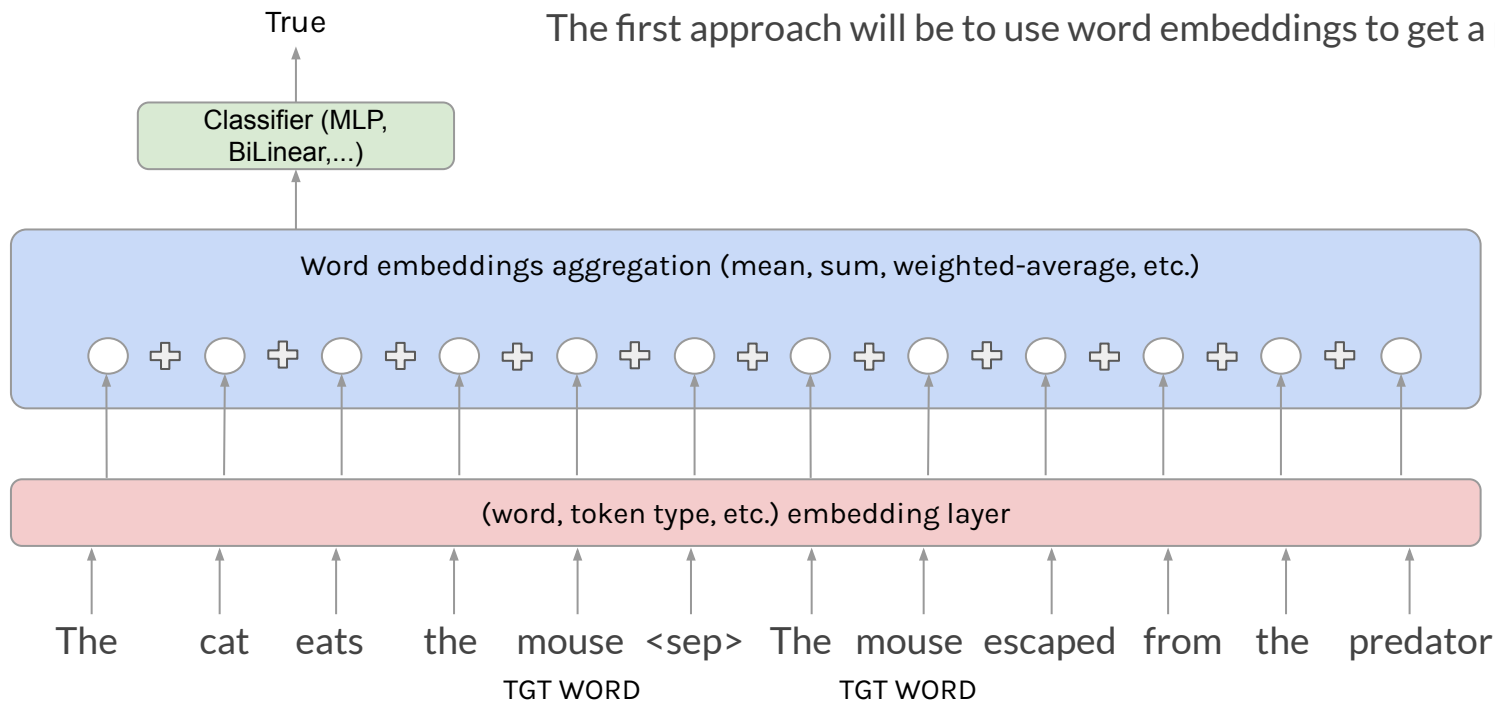


First approach (word-level)



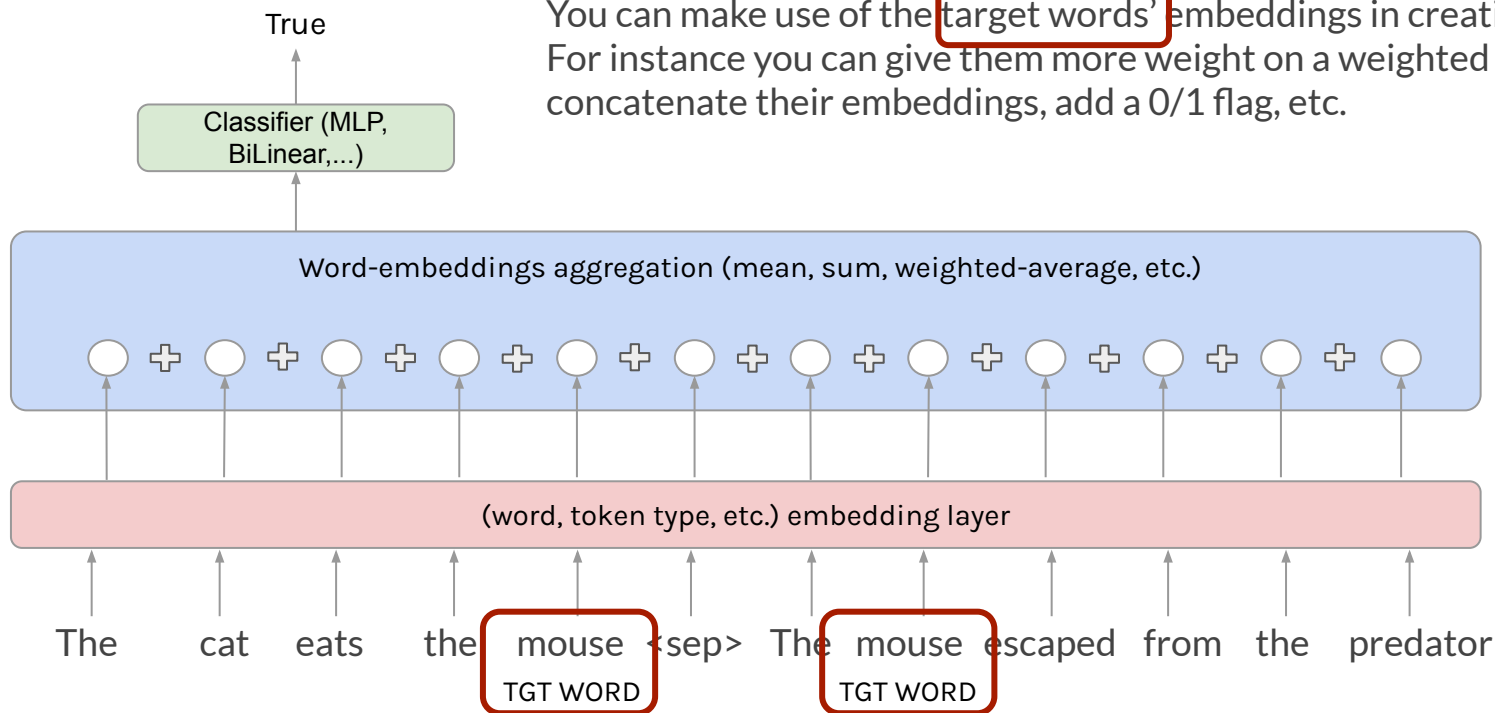
A simple approach to WiC

The first approach will be to use word embeddings to get a prediction.



A simple approach to WiC

You can make use of the **target words'** embeddings in creative ways. For instance you can give them more weight on a weighted sum, concatenate their embeddings, add a 0/1 flag, etc.

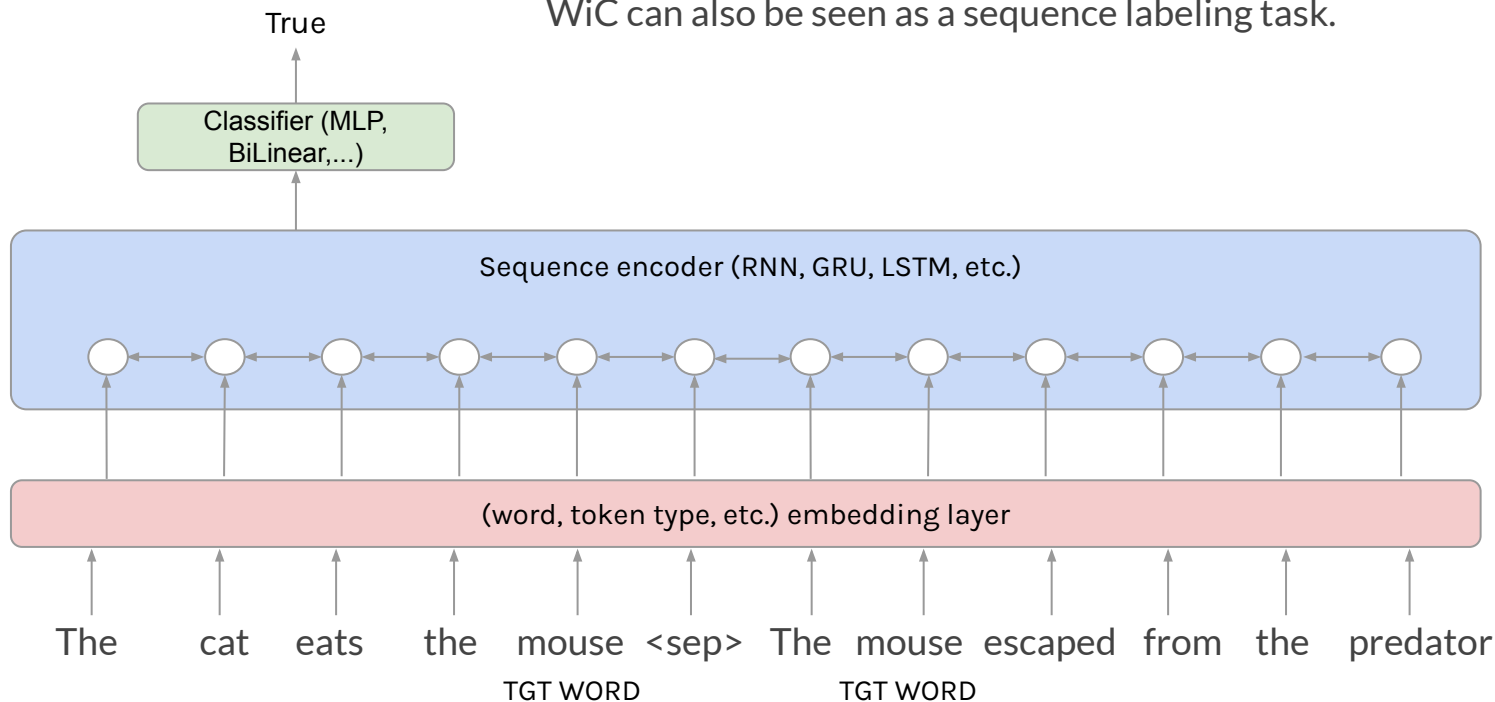


Second approach (sequence encoding)



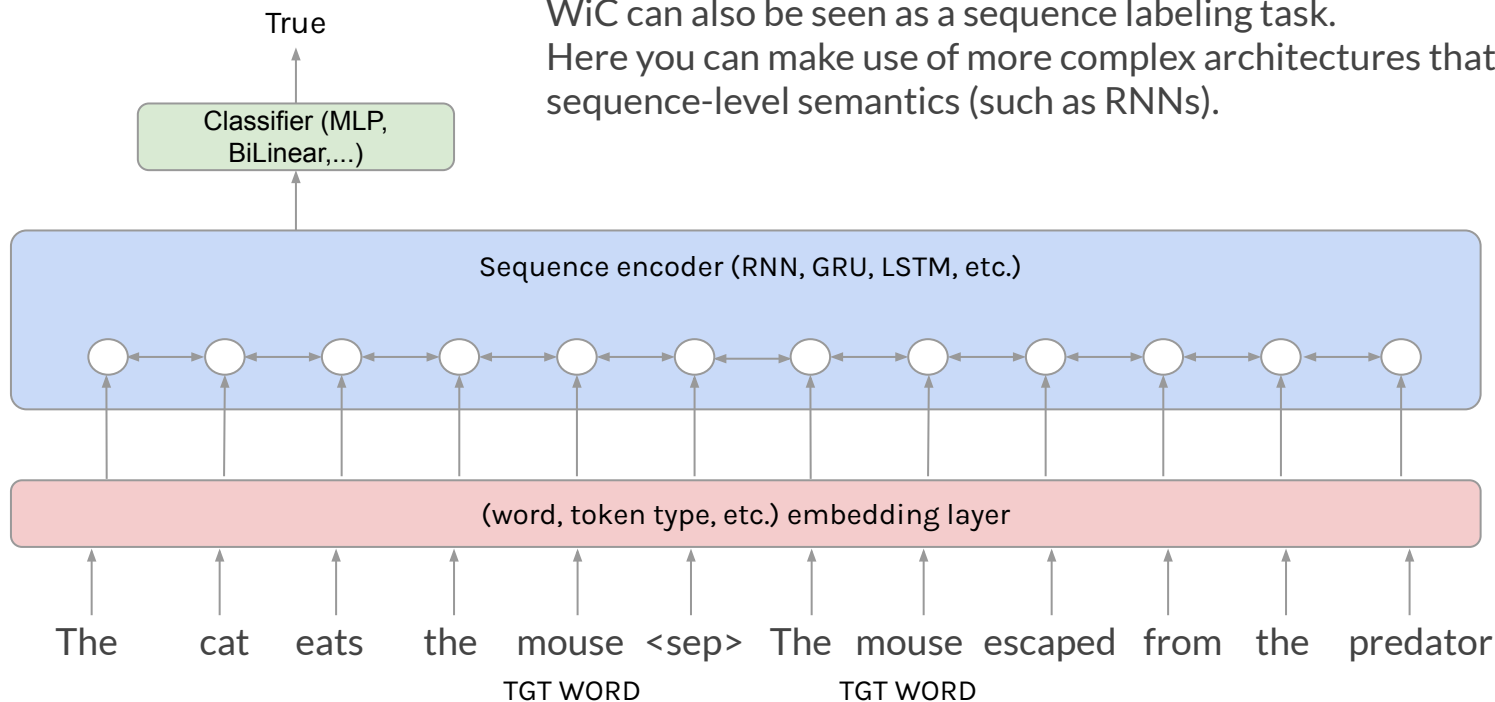
A sequence encoding approach to WiC

WiC can also be seen as a sequence labeling task.



A sequence encoding approach to WiC

WiC can also be seen as a sequence labeling task.
Here you can make use of more complex architectures that exploit sequence-level semantics (such as RNNs).

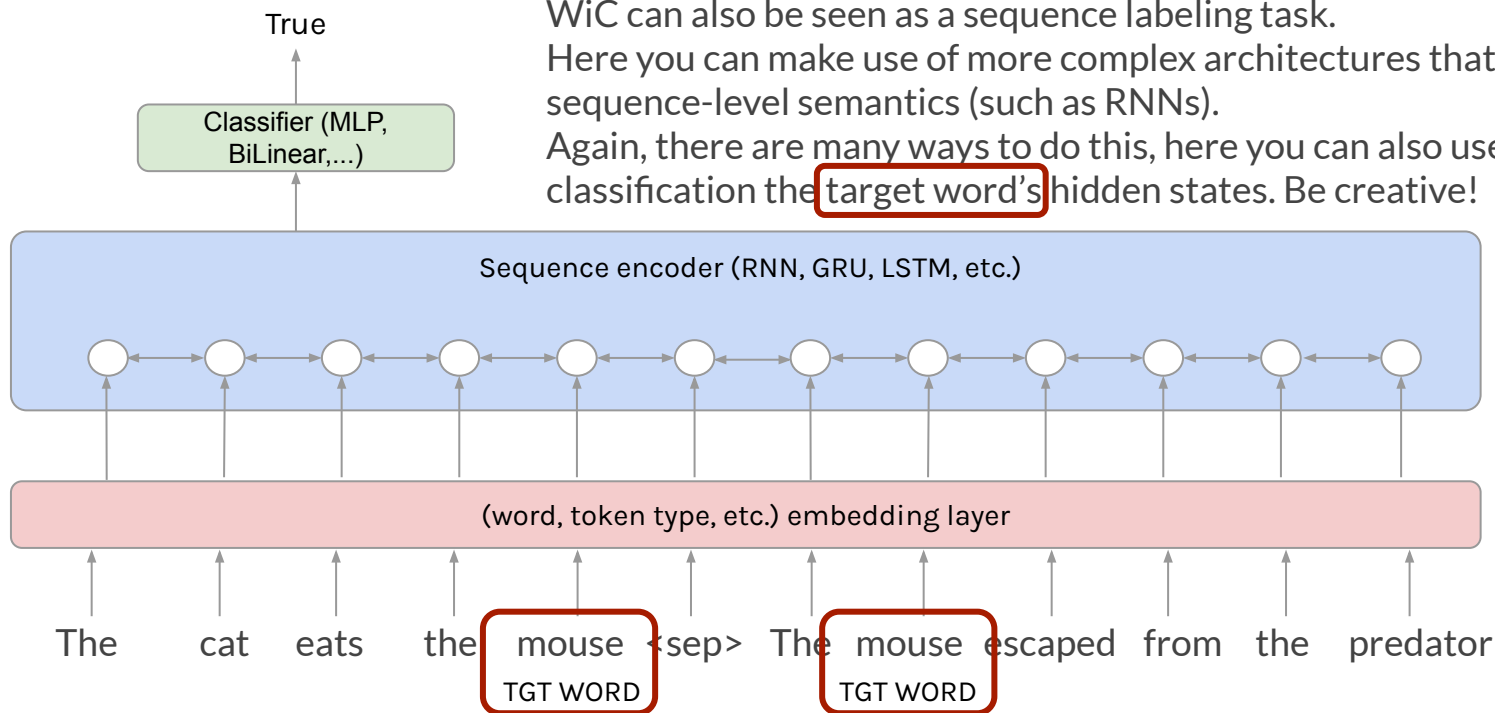


A sequence encoding approach to WiC

WiC can also be seen as a sequence labeling task.

Here you can make use of more complex architectures that exploit sequence-level semantics (such as RNNs).

Again, there are many ways to do this, here you can also use for classification the **target word's** hidden states. Be creative!



WiC: evaluation

The performance of a WiC system is usually measured in terms of accuracy.

GOLD	True	False	True	True	True	False
PRED	True	True	True	True	False	False

Accuracy = (CORRECTLY classified sentence pairs) / (TOTAL pairs) = 4/6 \approx 0.66



Submission

What you will receive &
How to submit

What you will receive: data format

The dataset is a JSONLINES file where each entry contains the following fields:

```
{  
  "id": "dev.24",  
  "lemma": "filthy",  
  "pos": "ADJ",  
  "sentence1": "They were given food in a filthy toilet without even minimal sanitary arrangements.",  
  "sentence2": "He spent a week in a filthy cell together with seven other detainees.",  
  "start1": "26",  
  "end1": "32",  
  "start2": "21",  
  "end2": "27",  
  "label": "True"  
}
```


What you will receive: data format

The dataset is a JSONLINES file where each entry contains the following fields:

```
{  
  "id": "dev.24",  
  "lemma": "filthy",  
  "pos": "ADJ",  
  "sentence1": "They were given food in a filthy toilet without even minimal sanitary arrangements.",  
  "sentence2": "He spent a week in a filthy cell together with seven other detainees.",  
  "start1": "26",  
  "end1": "32",  
  "start2": "21",  
  "end2": "27",  
  "label": "True"  
}
```

Input context sentences

What you will receive: data format

The dataset is a JSONLINES file where each entry contains the following fields:

```
{  
  "id": "dev.24",  
  "lemma": "filthy",  
  "pos": "ADJ",  
  "sentence1": "They were given food in a filthy toilet without even minimal sanitary arrangements.",  
  "sentence2": "He spent a week in a filthy cell together with seven other detainees.",  
  "start1": "26",  
  "end1": "32",  
  "start2": "21",  
  "end2": "27",  
  "label": "True"  
}
```

First char (included) and last char (excluded)
of target word in sentence 1

What you will receive: data format

The dataset is a JSONLINES file where each entry contains the following fields:

```
{  
  "id": "dev.24",  
  "lemma": "filthy",  
  "pos": "ADJ",  
  "sentence1": "They were given food in a filthy toilet without even minimal sanitary arrangements.",  
  "sentence2": "He spent a week in a filthy cell together with seven other detainees.",  
  "start1": "26",  
  "end1": "32",  
  "start2": "21",  
  "end2": "27",  
  "label": "True"  
}
```

First char (included) and last char (excluded)
of target word in sentence 2



What you will receive: data format

The dataset is a JSONLINES file where each entry contains the following fields:

```
{  
  "id": "dev.24",  
  "lemma": "filthy",  
  "pos": "ADJ",  
  "sentence1": "They were given food in a filthy toilet without even minimal sanitary arrangements.",  
  "sentence2": "He spent a week in a filthy cell together with seven other detainees.",  
  "start1": "26",  
  "end1": "32",  
  "start2": "21",  
  "end2": "27",  
  "label": "True"  
}
```

Gold label of this sample

What you will receive: data format

The dataset is a JSONLINES file where each entry contains the following fields:

```
{  
  "id": "dev.24",  
  "lemma": "filthy",  
  "pos": "ADJ",  
  "sentence1": "They were given food in a filthy toilet without even minimal sanitary arrangements.",  
  "sentence2": "He spent a week in a filthy cell together with seven other detainees.",  
  "start1": "26",  
  "end1": "32",  
  "start2": "21",  
  "end2": "27",  
  "label": "True"  
}
```

Additional info on the target words that
you can use in your model (not mandatory)

What you will receive: data format

We will provide you a folder with the following structure:

```
nlp2021-hw1/  
  data/  
  hw1/  
    model.py  
    stud/  
  model/  
  requirements.txt  
  test.sh
```

You are allowed to edit **only** the items in bold.

We will use Docker for evaluation. As far as you **do not change** any file but those we marked in bold, **if test.sh runs** on your side, it will run on ours as well. Find the code repository [here](#).

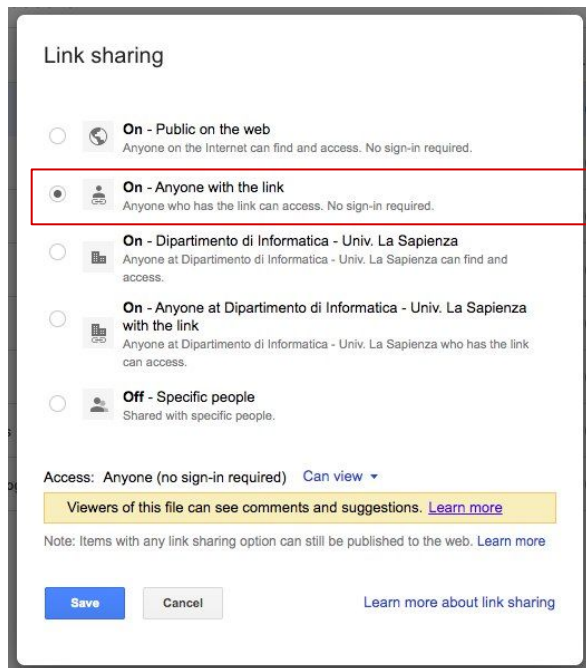
What we expect from you

- The zip folder we gave you (but populated :))
- Put your training code (if you used Colab, download the notebook **.ipynb** and place it) in **hw1/stud/**
- If you use any additional library, modify the requirements.txt file as needed (click [here](#) for info)
- Use the data (train, dev and test) in the data folder; **use each file as defined in the standard ML conventions** (train for training, dev for model selection, ...)
- Put everything your model needs (vocabulary, weights, ...) inside the **model/** folder, and be sure to properly load them in your model

What we expect from you


1. In `hw1/stud/implementation.py` implement the `StudentModel` class
 - Load your model and use it in the `predict` method
 - You **must respect the signature** of the `predict` method
 - You can add other methods (i.e. the constructor)
2. In `hw1/stud/implementation.py` implement the `build_model` function, initializing your `StudentModel` class.
3. Use `test.sh` to check that everything works
4. Add your `report.pdf` to the folder (yes, export it in pdf even if you are using Word!)
5. Name the zip folder `lastname_studentid_hw1.zip`
 - Ex: Luigi D'Andrea will submit a file named `dandrea_1234567_hw1.zip`


Submission instructions





The screenshot shows the 'Link sharing' settings for a Google Drive file. The 'On - Anyone with the link' option is selected and highlighted with a red rectangle. Below it, the 'On - Dipartimento di Informatica - Univ. La Sapienza' option is also visible. At the bottom, there are 'Save' and 'Cancel' buttons, and a link to 'Learn more about link sharing'.


Link sharing

☐  **On - Public on the web**
Anyone on the Internet can find and access. No sign-in required.

☒  **On - Anyone with the link**
Anyone who has the link can access. No sign-in required.

☐  **On - Dipartimento di Informatica - Univ. La Sapienza**
Anyone at Dipartimento di Informatica - Univ. La Sapienza can find and access.

☐  **On - Anyone at Dipartimento di Informatica - Univ. La Sapienza with the link**
Anyone at Dipartimento di Informatica - Univ. La Sapienza who has the link can access.

☐  **Off - Specific people**
Shared with specific people.

Access: Anyone (no sign-in required) [Can view](#) ▼

Viewers of this file can see comments and suggestions. [Learn more](#)

Note: Items with any link sharing option can still be published to the web. [Learn more](#)

[Save](#) [Cancel](#) [Learn more about link sharing](#)

- Upload the zip on your **institutional** Drive and make it **link-shareable** and **public to anyone** (an automatic script will download it).
- Make sure it is accessible via an incognito page of your browser!
- Do **NOT** modify the folder structure
- You have to submit the homework through the submission form on Google Classroom. You will be asked to fill a form with the requested information and the **link** to the zip you uploaded on Drive.



Evaluation

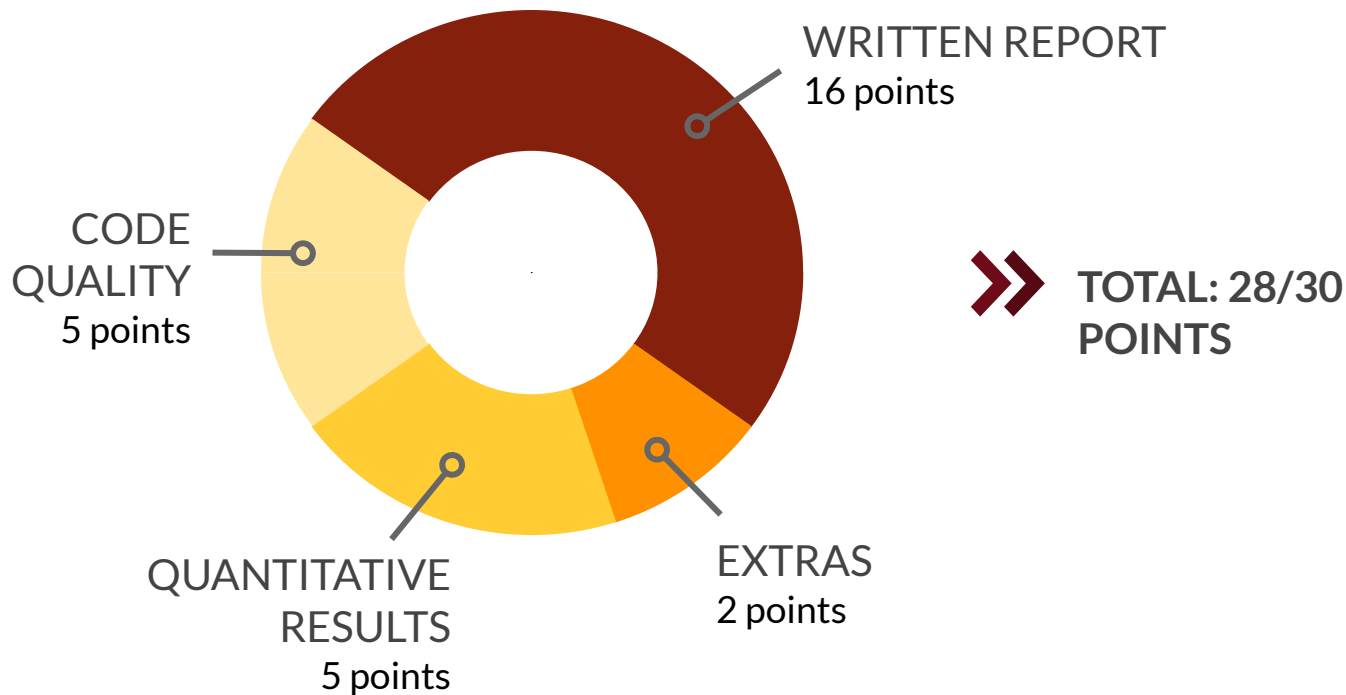
How your work will be evaluated



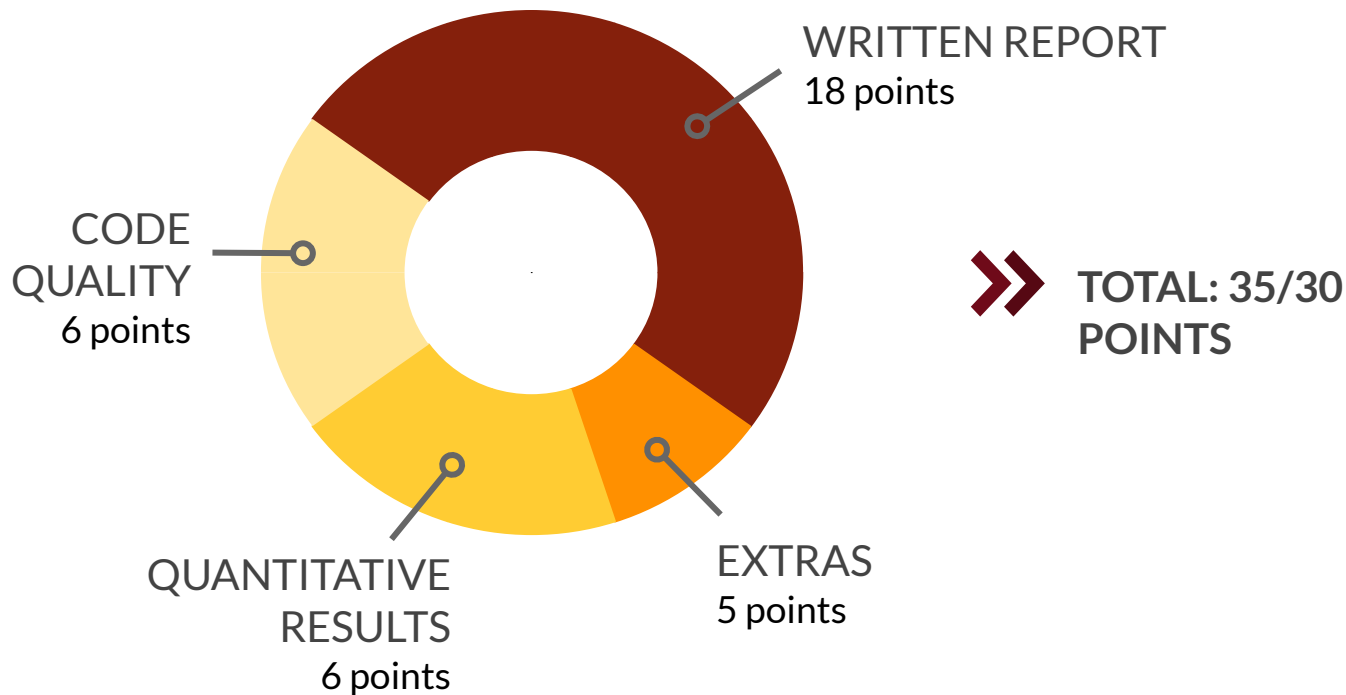
Homework 1 - Word-in-Context Disambiguation



Evaluation Overview (only first step submitted)



Evaluation Overview (second step included)



Report: dos and don'ts

- **ACL 2021 paper template**
 - Available [here](#) (Word and LaTeX direct download) or [here](#) (Overleaf LaTeX template)
 - You can use either the LaTeX or the Word template, your choice
 - **DO NOT MODIFY** the template (margins, spacing, font size)
 - Use the non-anonymous flag, so you can enter your name
- **Max 2 pages**
 - For the report, including title, subtitles, etc.
 - This is a **STRICT RULE!**
- **Unlimited extra pages for images, tables and references**
 - Every image and table must have a caption (don't abuse them please :-))
 - Tables and images must be referenced in the report

Report: what you are expected to do

We expect a good report to be:

- **Readable and understandable**
 - We will not give penalties for English errors, but we expect the report to follow a clear flow. We don't want to read just a sequence of statements on what you did without showing the reasoning behind your choices
- **Well-structured and organized**
 - Take inspiration from the many papers available online and organize your report in well-defined sections (e.g. method, setup, experiments, results...)

Report: what you are not expected to do

We expect a good report **NOT** to include:

- Unnecessary **task** or **dataset descriptions**
 - just focus on your solution to the problem.
- **Code** copy-paste
 - Your code should be self-explanatory, so no need to show it in the report. You can add **pseudo-code** to show some particular algorithm, but **no code or screenshots** please!

Report: what you are not expected to do

We expect a good report **NOT** to include:

- Unnecessary **low-level implementation details**
 - Avoid any **low-level implementation/technical details** like “I used a dictionary to store these values”, “I had to use configuration X to solve this exception”, “I could not use Y because there was a dependency issue with Z”, etc.
 - Instead, **we are interested in high-level abstractions/strategies** you decide to use to tackle the homework, as well as the **intuitions behind your choices**.
E.g. use and description of a particular model, explanation of how and why an architecture works, etc.

Code and code Quality

Your project should conform to the following rules:

- You **MUST** use PyTorch.
 - TensorFlow and other deep learning frameworks are **NOT** allowed.
 - PyTorch Lightning is **NOT** allowed (at this stage)
- **Frameworks** that use PyTorch (e.g. AllenNLP, torchtext...) are **NOT** allowed.
 - Libraries (such as tqdm, sklearn, NLTK) are fine, but since the line between a framework and a library is sometimes blurred, please ask in the Google Classroom group before using any external library: **any other library MUST be agreed with the TAs.**



Code and code Quality

Your project should conform to the following rules:

- **You are not allowed** to use tools/architectures that have not been explained **yet** in the course, in particular:
 - word embeddings (Word2Vec, GloVe, etc.) **are allowed**,
 - contextualized word embeddings (ELMo, etc.) **are NOT allowed**,
 - Transformer-based models (BERT, BART, RoBERTa, XLM, etc.) **are NOT allowed**.
- For any doubt, please ask the TAs on Google Classroom.
- **Comment** your code, please!

Quantitative Results

We will evaluate the **performance of your model** on a SECRET test set.

You can get **from 0 to 6** points according to the following **thresholds**:

- $P < 0.52$ \Rightarrow FAIL
- $0.52 < P < 0.55 \Rightarrow 0$
- $0.55 < P < T2$ $\Rightarrow 1$
- $T2 < P < T3$ $\Rightarrow 2$
- $T3 < P < T4$ $\Rightarrow 3$
- $T4 < P < T5$ $\Rightarrow 4$
- $T5 < P < T6$ $\Rightarrow 5$
- $P > T6$ $\Rightarrow 6$

Quantitative Results

We will evaluate the **performance of your model** on a SECRET test set.

You can get **from 0 to 6** points according to the following **thresholds**:

- $P < 0.52$ \Rightarrow FAIL
- $0.52 < P < 0.55$ \Rightarrow 0
- $0.55 < P < T2$ \Rightarrow 1
- $T2 < P < T3$ \Rightarrow 2
- $T3 < P < T4$ \Rightarrow 3
- $T4 < P < T5$ \Rightarrow 4
- $T5 < P < T6$ \Rightarrow 5
- $P > T6$ \Rightarrow 6

Thresholds will be defined based on an internal reference model and the **normalized distribution of YOUR scores!**



Extras

You can achieve **up to 5 points with some extras!**

An “extra” is whatever you decide to add to your model to make it better. For instance:

- use of pre-trained embeddings,
- use of NLP best practices,
- comparative analysis of results in your report,
- informative plots in your report,
- **new ideas** (including using external resources in a clever way, please see slide 50)

and more, according to internal baselines. Don't forget to **explain your choices** in the report!
Extras that are not explained in the report will not be considered for evaluation.

Evaluation

- `test.sh` is identical to what we will be using
- If it does not run on your side, we will not correct your homework
- Note that, if you use **any kind of hard-coded paths**, this script won't work
- Use **paths relative to the project root folder**, e.g.:
 - **NO:** `/home/pincopallino/my_folder/model/weights.th`
 - **OK:** `model/weights.th`

Warnings

Things you should be aware of



Please be aware that

This is an **individual homework**! Collaboration among the students is **not** allowed.

We will check for **plagiarism** both manually and automatically.

It is **not allowed** to:

- Copy from other students
- Share your code with other students
- Copy from online resources (StackOverflow, GitHub, Medium and so on).

However, you are allowed to use material from **external sources** as long as it is **not central** to the homework.

- In this case, it is **MANDATORY** to cite such resources in the report

Please be aware that

- If we find out that you breached any of the above rules, you will **automatically FAIL** this homework and you will have to pass a **FULL EXAM**.
- **Plagiarism will imply further consequences at the Faculty level.**
- While we release the homework on GitHub, **DO NOT FORK THE PROJECT**.
- If you want to continue using GitHub for versioning, clone the project and re-upload it.
- If we realize you shared your code in any way (forking or otherwise), even without the intention of letting others copy, you will be failed automatically.

Use of external data

- For your experiments, **use the provided data** (train and dev) in the data folder; use each file as defined in the standard ML conventions (train for training, dev for model selection).
- **Use only the training set to train the model that you submit for evaluation.** If you train it on more data (dev set or any other external data), it will be a **FAIL**.
- You can use external data, but only as an **extra submission** in order to get extra points. If you do use external data for this purpose, make it clear in the report and submit both versions of the model so we can evaluate on the test set.
- **The model trained on extra data will only count towards extra points and it is not mandatory.** Moreover, it is not mandatory to train on extra data to get extra points.

Tips

A few tips to organize your work:

- Start as soon as possible!
 - Training a neural network requires time, possibly hours, depending on your hardware
- Start **small**!
 - If you don't get decent results with a very very simple neural network, there is a good chance that adding other things won't make your model perform better
 - Leave the “extras” as the last thing!
- Leave some time for **hyperparameter** tuning!
 - Sometimes good hyperparameter combinations can do wonders for your neural network
- Use Google [Colab](#) (free GPUs!)



Deadline

When to deliver what



Homework 1 - Word-in-Context Disambiguation

Deadline

Early submission date: **May 2nd, 2021**
23:59:59 Italian time (UTC + 1)

Submit the homework through the submission form on Google Classroom. You have to fill the form with the requested information and a link to the zip folder of the homework on Google Drive.

Late submission policy

Late submission date: May 7th, 2021.

1 point penalty will be applied for each day of delay, e.g.:

- A student delivers their homework on May 5th (17:35) -> max possible grade $35 - 3 = 32$
- A student delivers their homework on May 8th (17:35) -> FAIL!



Awards

Get a **Sapienza NLP™** t-shirt



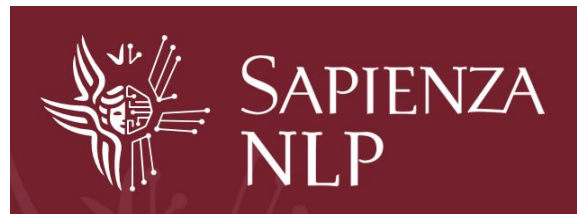
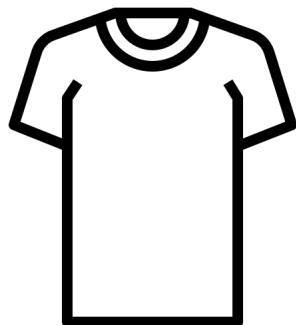
Homework 1 - Word-in-Context Disambiguation



Win a Sapienza NLP t-shirt!

We will hand out amazing Sapienza NLP t-shirts to the **overall top-5** students!

The final ranking will be computed according to the scores on our **secret** test set.



That's not all

If your work is novel, interesting and original, we will gladly invite you to work together with us to extend on a fully-fledged paper for **TOP-TIER INTERNATIONAL CONFERENCE!**

Just over the last 12 months, the Sapienza NLP group published more than a dozen of papers!

Questions?

If you have a question that may interest your colleagues, **please ask it on [Google Classroom](#)**.

Otherwise, for personal or other questions, send an email to **ALL** of us (but please, only reach for things that can't be asked on the Google Classroom).

Our emails are:

campagnano@di.uniroma1.it

huguetcabot@babelscape.com