Homework 2

Word Sense Disambiguation

(scozzafava | pyatkin)@di.uniroma1.it

What you will do:

- Implement a Word Sense Disambiguation System
  - Neural
  - Knowledge-Based
- The type of architecture is up to you! Be creative!

What we will provide:

- **Training Data:**
  - Origin: Semcor 3.0, processed by "Alessandro Raganato, Jose Camacho-Collados and Roberto Navigli. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison."
  - **Files:** semcor.data.xml, semcor.gold.key.bnids.txt
- **Development Data/Test Data:**
  - All the SenseEvals and SemEvals, processed by "Alessandro Raganato, Jose Camacho-Collados and Roberto Navigli. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison."
  - **Files:** ALL.data.xml, ALL.gold.key.bnids.txt
- **Final Test Data**:
  - A special dataset created by us
  - **Files:** finaltest.txt, we will upload the final test data about a week before the deadline.

Format of Data:

- FILE 1 :
  - XML
    - you have to parse it
  - Given: Lemma, POS, Token, instance ID
- FILE 2 :
  - Contains sense annotations
  - For a word in a sentence, get the instance id and look it up in this file
  - The sense annotations are BabelNet synsets.

```
-<sentence id="d000.s000">
   <wf lemma="how" pos="ADV">How</wf>
   <instance id="d000.s000.t000" lemma="long" pos="ADJ">long</instance>
   <wf lemma="have" pos="VERB">has</wf>
   <wf lemma="it" pos="PRON">it</wf>
   <instance id="d000.s000.t001" lemma="be" pos="VERB">been</instance>
   <wf lemma="since" pos="ADP">since</wf>
   <wf lemma="you" pos="PRON">you</wf>
   <instance id="d000.s000.t002" lemma="review" pos="VERB">reviewed</instance>
   <wf lemma="the" pos="DET">the</wf>
   <instance id="d000.s000.t003" lemma="objective" pos="NOUN">objectives</instance>
   <wf lemma="of" pos="ADP">of</wf>
   <wf lemma="you" pos="PRON">your</wf>
   <instance id="d000.s000.t004" lemma="benefit" pos="NOUN">benefit</instance>
   <wf lemma="and" pos="CONJ">and</wf>
   <instance id="d000.s000.t005" lemma="service" pos="NOUN">service</instance>
   <instance id="d000.s000.t006" lemma="program" pos="NOUN">program</instance>
   <wf lemma="?" pos=".">?</wf>
 </sentence>
```

```
d000.s000.t000 1437963a
d000.s000.t001 2604760v
d000.s000.t002 696189v
d000.s000.t003 5981230n
```

Sense-annotated training corpora

- **SemCor** (Miller et al., 1994)

  Semcor is a manually sense-annotated corpus divided in **352 documents** for a total of **225,040 sense annotations**. SemCor is originally tagged with senses from the WordNet 1.4 sense inventory and is the main corpus used in the literature to train supervised WSD systems.

- **OMSTI** (Taghipour and Ng, 2015a)

  The ***One Million Sense-Tagged Instances*** is a large corpus annotated with senses form WordNet 3.0 inventory. It is automatically constructed by using an alignment-based WSD approach on a large English-Chinese parallel corpus. OMSTI has been used to improve the performances of supervised systems which add it to existing training data.

WSD Evaluation Datasets

- **Senseval-2** (Edmonds and Cotton, 2001)
    - Originally annotated with WordNet 1.7 consists of **2282 sense annotations**, including nouns, verb, adverbs and adjectives.
- **Senseval-3 task 1** (Snyder and Palmer, 2004)
    - Originally annotated with WordNet 1.7.1, consists of three documents from three different domains (editorial, news story and fiction), totaling **1850 sense annotations**.
- **SemEval-07 task 17** (Pradhan et al., 2007)
    - This is the small among the five datasets, it contains **455 sense annotations** for nouns and verbs only. Annotated using WordNet 2.1 sense inventory.
- **SemEval-13 task 12** (Navigli et al., 2013)
    - This dataset includes 13 documents from various domains. Annotated with WordNet 3.0, it includes **1644 sense annotated nouns**.
- **SemEval-15 task 13** (Moro and Navigli, 2015)
    - The most recent WSD dataset available to date, annotated with WordNet 3.0. It consists of **1022 sense annotations** in four documents coming from heterogeneous domains.

An example of Neural WSD system

Neural Sequence Learning Models for Word Sense Disambiguation

(Raganato, Delli Bovi, Navigli, EMNLP17)

- Supervised approach for the all-words WSD task
- The authors present three approaches
    - Supervised bi-LSTM based approach
    - Supervised bi-LSTM based approach with attention
    - Sequence-to-Sequence encoder-decoder architecture

Neural Sequence Learning Models for WSD

- Input: variable-length sequence of input symbols $\vec{x} = \langle x_1, ..., x_T \rangle$
- Output: variable-length sequence of output symbols $\vec{y} = \langle y_1, ..., y_{T'} \rangle$

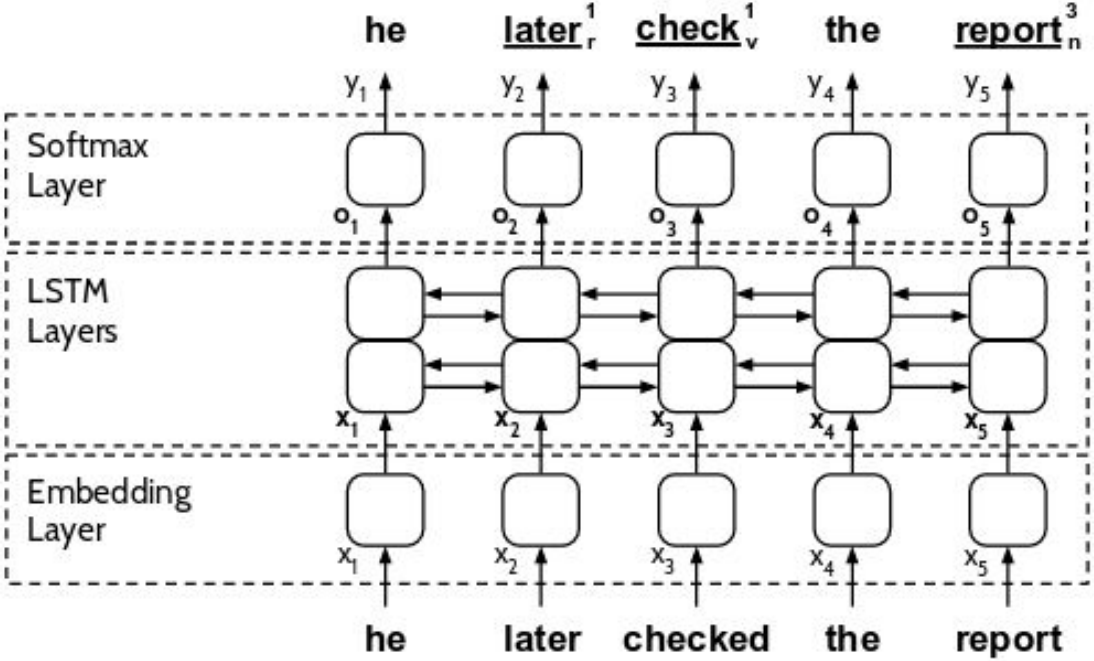
Input symbols are word tokens drawn from a given vocabulary **V**, output symbols are either drawn from a pre-defined sense inventory **S** or from the same input vocabulary **V**.

Bidirectional LSTM Tagger

Architecture:

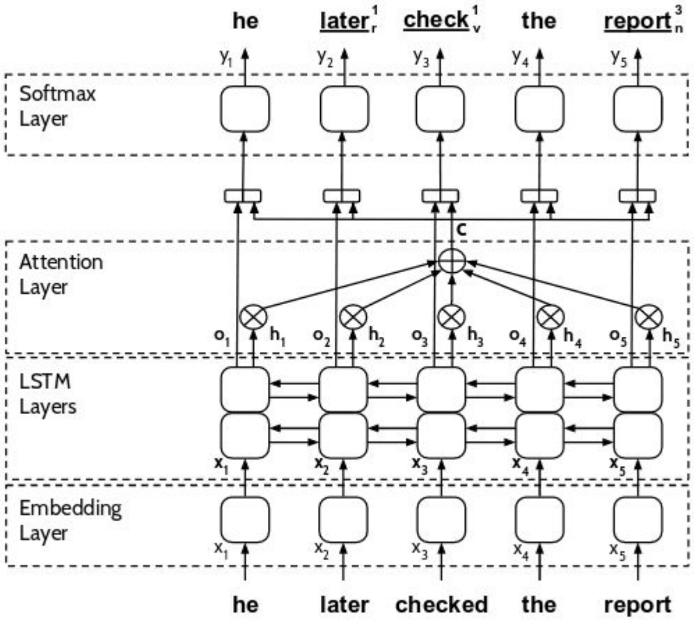
- Embedding layer that converts each word $x_i \in \vec{x}$ into a real-valued d-dimensional vector
- One or more stacked layers of bi-LSTM
- The hidden and output states are obtained by concatenating bw and fw states
- Fully-connected layer with softmax activation which produces a probability distribution over the output vocabulary at each time step

Bidirectional LSTM Tagger



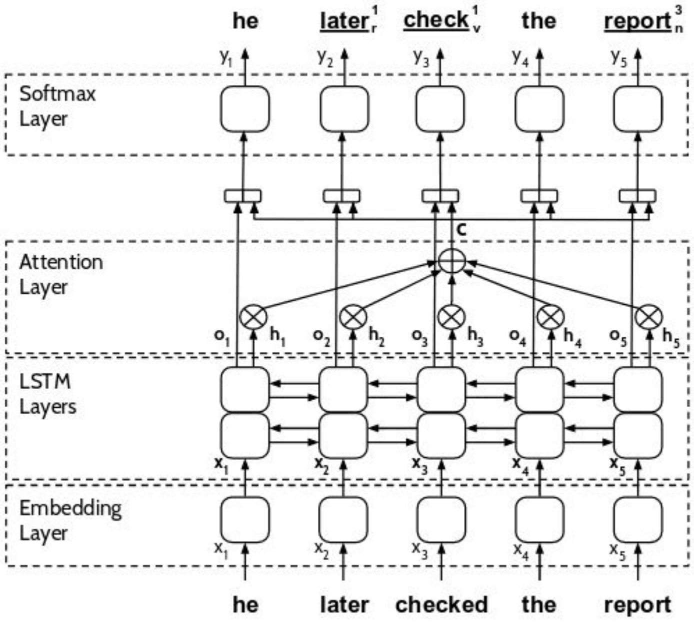
Attentive Bidirectional LSTM Tagger

While the simple **bi-LSTM** tagger exploits information from the input sequence, the introduction of the **attention mechanism** makes the system able to learn which elements are more **discriminative in predicting the output label** at a given time step.



Attentive Bidirectional LSTM Tagger

- **Two-pass** procedure
- The context vector **c** is computed from all the hidden states
- **c** is concatenated to the output vector for each time step and it's exploited to predict the output label



Attention mechanism

Formally, the attention vector **c** is defined as follows:

$$\mathbf{u} = \omega^T \tanh(H)$$
$$\mathbf{a} = softmax(\mathbf{u})$$
$$\mathbf{c} = H\mathbf{a}^T$$

$H \in \mathbb{R}^{n \times T}$ is the matrix of the hidden states, $\omega \in \mathbb{R}^n$ is the parameter vector and

$\mathbf{a} \in \mathbb{R}^T$ is the vector of normalized attention weights.

Sequence-to-Sequence model

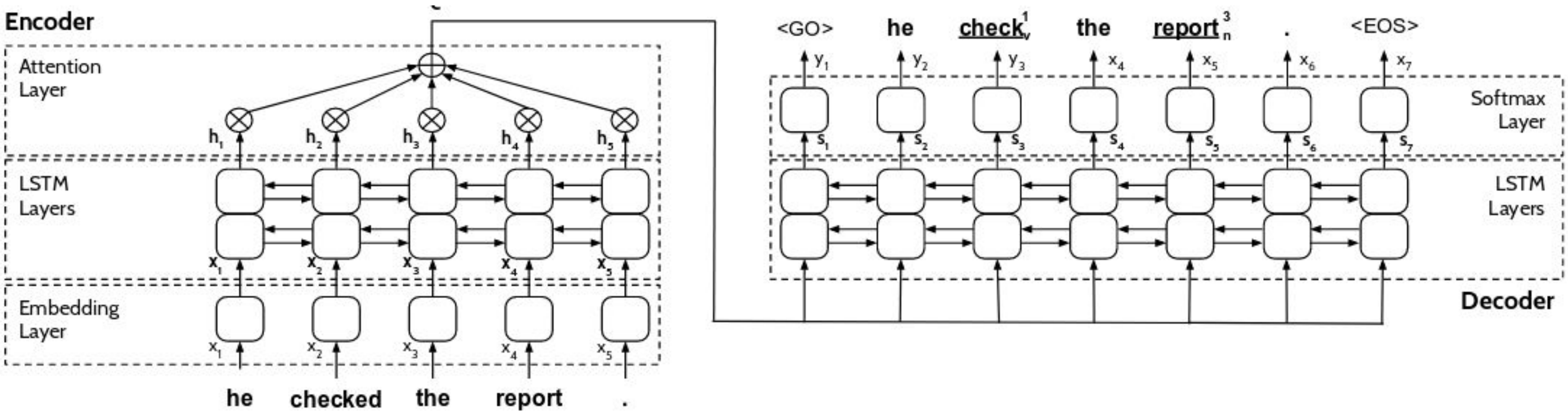
The attentive model performs two-pass procedure by first reading the input sequence ***x*** to construct the context vector ***c***.

The attentive architecture can be viewed as an encoder for the sequence *x*.

- The vectors are fed into an encoder which generates a fixed-dimensional vector representation of the sequence
- The decoder is trained to predict the next output symbols ***y***

Sequence-to-Sequence architecture

- **Encoder**:
  - Construct the context vector ***c*** (last hidden state or attentive layer)
- **Decoder**:
  - Takes ***c*** as input
  - bi-LSTM layers generates the output sequence
  - ***c*** is provided in input to the LSTMs at every timestep



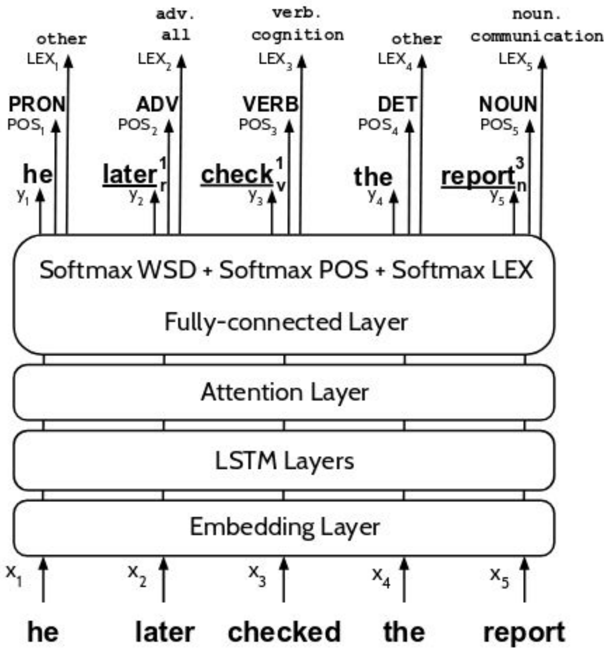
Extra: Multitask Learning!

MTL consist in a single architecture trained using **multiple loss functions** and a **shared representation:**

- One task-specific output layer per additional task
- Usually **located at the outermost level** of the architecture
- The remaining hidden layers are **common across all tasks**
- The intention is to **improve the main task** by jointly learning more related auxiliary tasks
- WSD is strongly linked to other NLP tasks at various levels:
  - Part Of Speech
  - Coarse-grained semantic labels

Multitask Learning Architecture

Each output is handled by a dedicated output layer



Models

Supervised:

- Follow one of the proposed architectures
- Feel free to combine ideas and different models
- **Ideas for novelty:**
    - Modify/improve one of the proposed architectures with auxiliary input/outputs
    - Exploit semantic embeddings (see http://lcl.uniroma1.it/sensembed/)
    - Integrate information from a Knowledge-base (BabelNet) into the network
        - For example by developing MultiTask learning
        - By integrating sense embeddings / synset embeddings / multilingual word embeddings

Ideas for Novelty: Knowledge-Based

Knowledge-Based:

- Create a semantic graph of the document including all the synsets extracted from the words
  - Use the sense embeddings to create a centroid
- Graph heuristics
  - Personalised PageRank on the graph
  - Find the most connected subgraph

Ideas for Novelty: Addressing the Knowledge Acquisition Bottleneck for WSD

- Find a new strategy to extract new semantically-tagged training sentences for the words in the test sets
- Improve the disambiguation performance in supervised and/or knowledge-based approaches by leveraging the acquired sentences
- How could you create new training data? Ideas:
    - Re-implement a version of: Pasini, Tommaso, and Roberto Navigli. "Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data." *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.
    - Project annotations across languages or train a cross-lingual model.

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 [**35%**]
  - Report [**40%**]
  - Novelty [**20%**]
  - Overall performance of the system [**20%**]
  - In order to get 30+ you have to obtain an F-measure score higher than the baseline defined below for each kind of system
  - The **MFS baseline** are defined as follows:

	Senseval2	Senseval3	SemEval07	SemEval13	SemEval15
Baseline Sup.	65.6%+4	66.0%+4	54.5%+4	63.8%+4	67.1%+4
Baseline KB	65.6%-2	66.0%-2	54.5%-2	63.8%-2	67.1%-2

What you have to submit

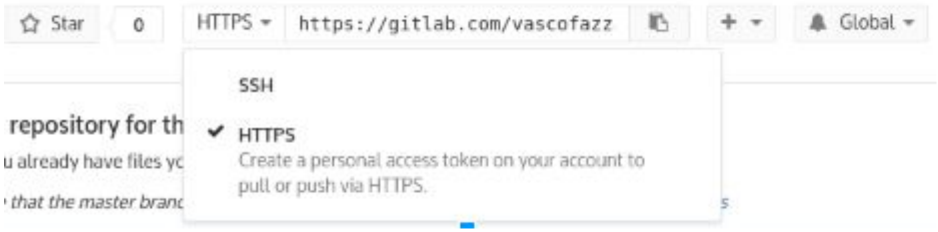
- FILL OUT THE **GOOGLE FORM** (MANDATORY!)
- You will be prompted to submit the following:
  - Your **report** (max. 1 page of written text with unlimited pages for graphs/tables and other eventual appendices)
  - Your **answers** on the test data: **<matricola>_test_answer.txt** **(es: 1234567_test_answer.txt)**
    - **FORMAT**: **id\tanswer\n** **(id TAB answer NEWLINE)**
    - **For example: d000.s000.t000\tbn:00005054n\n**
- Your code in a gitlab repository (permission set to 'developer')
  - The gitlab repository has to **only** contain your source files and your code!

Report

- Explain the architecture of your system
  - Give as much details as possible about Preprocessing and the implementation
- Mention all your results on the SemEvals/SenseEval and compare them to the baseline (eventually with graphs and/or tables)
- Choose good and bad examples and try to reason about them, see if you can find any patterns or mention other impressions you have.

Source code submission

- Register to GitLab.com and create a new project
- Name the project **firstname_lastname_matricola_nlp18hw2**
- Share the project with (project setting -> members):
  - pyatkin@di.uniroma1.it
  - federico.scozzafava@gmail.com
  - navigli@di.uniroma1.it
- **Give us developer permissions**
- Upload your code on the repository
- Get the **HTTP URL**



Deadlines

- We will upload everything you need to complete this homework Sunday evening the latest, on the facebook group
- Your deadline for homework 2 will be: **03.06.18, 24:00 Pacific Time + Fibonacci for late deliveries**
- Link to the **submission** GOOGLE FORM

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

Good Luck!

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