[Reflections and the next steps – 16/10 and onwards 2](#_Toc22202003)

[Development – 17/10 and onwards 5](#_Toc22202004)

[Step 0 : Vocabulary 5](#_Toc22202005)

# Reflections and the next steps – 16/10 and onwards

Add-on: are there any related papers incoming from EMNLP 2019?

* “Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations” by Christian Hadiwinoto, Hwee Tou Ng and Wee Chung Gan
* “Challenging Supervised Word Sense Disambiguation with Lexical-Semantic Combinations” by Marco Maru, Federico Scozzafava, Federico Martelli and Roberto Navigli
* “Towards Zero-shot Language Modelling” by Edoardo Maria Ponti, Ivan Vulić, Ryan Cotterell, Roi Reichart and Anna Korhonen
* “MultiFiT: Efficient Multi-lingual Language Model Fine-tuning” y Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kadras, Sylvain Gugger and Jeremy Howard
* “GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge” by Luyao Huang, Chi Sun, Xipeng Qiu and Xuanjing Huang
* **Read** references on multi-sense embeddings & language models  
  90%+ done.   
  A quick overview:
  + J. Resinger & R.J. Mooney 2010: represent the words & contexts with TF-IDF / χ-square features, and then senses = centroids of the context clusters
  + Eric H. Huang et al. 2012: local+global(document) FF-NN architecture. Single-prototype phase: Quasi-LM, I see the whole document, but the objective is a score on the next word.  
    Then, cluster the contexts with spherical k-means.
  + X. Chen et al. 2014:   
    Single-prototype phase: Skip-gram  
    Senses from WordNet: definitions and examples form ‘glosses’; we take the average of the candidate vectors in glosses to get the sense embeddings.  
    WSD is done using the cosine-sim between context window and sense.
  + S. Rothe & H. Schutze 2015: AutoExtend  
    Embeddings for synsets and lemmas, using an AutoEncoder. Exploits the linearity assumption of a synset being constructed as the sum of lemmas.  
    Method: word embeddings (possibly pre-trained) -> encode -> synset embeddings -> decode -> re-create word embeddings
  + A. Neelakantan et al., 2015: MSSG, Multi-Sense Skip-Gram model  
    Single-prototype phase: initial global (Skip-Gram) word vectors.  
    2 variants. One with a fixed number of senses, the other that determines the senses based on the locations of the context embeddings (NP-MSSG)
  + S.K. Jauhar et al., 2015: Interacts with an ontology, uses context-independent sense proportions, considering the senses as latent variables
  + J. Li & D. Jurafsky 2015:   
    Uses CRP, Chinese Restaurant Process to create and populate new senses with contexts.  
    WSD can be either greedy or expectation.
  + H. Shi et al., 2016: Exploration of Pseudo Multi-Sense
  + D.Kartsaklis et al., 2018: Mapping text to KB entities  
    Only relevant here because it uses a (Q,K,V) attention mechanism on a fixed number of vectors (k=3) to choose the senses (with soft-attention) and update them.
  + M.T. Pilehvar et al., 2017: Given a text T, turn it into a graph (S,E) of all possible Senses + relation Edges.  
    At each iteration: select the sense s\_ij with the highest degree in the graph.
  + I. Iacobacci and R. Navigli, 2019: LSTMembed  
    1. Obtain a sense-labeled corpus, using the Babelfly instrument   
    2. BiLSTM over the training text > 1 layer > Training objective: minimize the cosine distance from a pre-trained embedding
  + A. Panigrahi et al., 2019: Word2Sense: interpretable embeddings  
    The dimensions have meaning:   
    A sense is a concept, represented by a set of similar words in the vocabulary. Like topics, senses are distributions over the vocabulary.   
    k senses = k dimensions; a word w will have a score on each sense.  
    (3000 -> JS-merge into ¾ -> truncate to 75 eventually)
  + S. Kumar et al., 2019: EWISE  
    Encode context with BiLSTM, then project…   
    Training objective: minimize the cosine distance from the pre-existing sense embeddings, that were created independently, either from pre-trained models, or from KB (e.g. WordNet) resources
* **Read** on evaluation measures for multi-sense embeddings & language models  
    
  In the works I consulted (most of which were mentioned in the previous section), the evaluation measures that were used most often and/or most promisingly were:
  + SCWS (Stanford's Contextual Word Similarities).  
    Better than basic word similarity tasks such as WS-353 or RG-65; it also allows us to apply the WSD mechanism
  + {Downstream tasks: PoS-tagging, SentEval}
  + Downstream task: STS, Semantic Textual Similarity (STS12, STS13, STS14, STS15, STS16, STS-B), and also the SICK relatedness (SICK-R) and SICK-E (where the classes are ‘entailment’, ‘contradiction’, and ‘neutral’)
  + Word Sense Disambiguation: in particular, SemEval 2013 and 2015
  + Finally, Downstream task: Language model  
    WikiText-2 as a small, pocket-sized development set.  
    WikiText-103 as a larger dataset. We should obtain comparable/acceptable perplexity.

Note: it is also opportune to study the impact of dimensionality   
(e.g. 300, 384, 768)

* **A baseline** Multi-Sense LM  
  What models are going to compare our results with?   
  With different models, depending on the selected task. For some of them, When operating on well-known datasets, we do not need to train & implement anything, we can just compare result scores.  
  It is also opportune to consider models that are not too unorthodox (e.g. Pseudo Multi-sense, or interpretable Word2Sense) and not too old (Huang et al. 2012, FF-NN + context-based) .  
  Hypothesis regarding the models:
  + X. Chen et al. 2014. It uses Skip-gram > candidate words in the glosses > vectors for senses.  
    It is an approach relatively close to mine, but without any GNN.
  + A. Neelakantan et al., 2015: MSSG, Multi-Sense Skip-Gram. Here the senses are obtained from the contexts. A generic Multi-sense instrument, examined often and good for comparisons.
  + Transformer-XL, 2019. It already has perplexity values on WT-2 and more importantly WT-103. I even introduced it as a baseline (and a future feature) for Write-Assistant.
  + Possibility: The (pre-trained, unmodified) BERT.  
      
    We mention again here the (candidate) evaluation tasks:
  + SCWS for word similarity.
  + STS (Semantic Textual Similarity on sentences)
  + Language Modeling perplexity, on WT-2 and WT-103
  + WSD

# Development – 17/10 and onwards

* **Develop** the Graph-Based Multi-Sense Word Embeddings:
  + Step 0: define a vocabulary
  + Observation: our dictionary-resources system can only deal directly with base, non-inflected forms of words (e.g. only ‘plant’, not ‘plants’ or ‘planted’)
    - Lemmatization for inflected forms when training, to include them in the processing
    - We can worry about this later, in the next phase. Several alternative solutions are possible:
      * Inflected forms can be adjusted using a standard corpus-based approach
      * We can add singular / plural / past edges to the graph
      * We may also concatenate dict-vecs for base forms, and context-based vecs for inflected forms
  + Preliminary distributional embeddings from training corpus, in order to provide for words without a dictionary definition (for, of, etc.). This also initializes the 'move.global' vectors.
  + Placing in the multi-dimensional space the definitions, and the examples. Sentence embedding based on BERT (average over the word tokens of the 2nd-to-last layer)
  + What about the embedding dimension? I can try:  
    - the original bert-base-uncased size (768)   
    - PPA+PCA+PPA from V.Raunak et al. 2019 (768 / 2 = 384, or even to 300)  
    - PCA to 300
  + Consider the senses (move.v.1, ..., move.n.3) from the KB sources. How to initialize them? Alternatives:   
    1) on move.global, plus a random perturbation  
    2) Average of all definitions and examples for that particular sense
  + Build Graph Neural Network (Graph Attention Net, or DCGCN)
  + Language Modeling objective on the training set.
  + Evaluate (see previous discussion)
  + The evaluation should be comparable with:  
    - SoA models on downstream tasks (eg. LSTMs for LMs on WT-2, we should not be too far off)  
    - contextual Transformer embeddings.  
    - And also old Multi-sense embeddings (MSSG, C.R.P.)

## 

## Step 0 : Vocabulary

Our vocabulary must be useful for Language Modeling.

* Word-level or subword level? Word level.  
  BERT uses internally subword level, but we employ it only to produce sentence embeddings for defs & examples.
* Keep punctuation or remove it? Keep it. Later on, possibly, a version for Write Assistant may remove it.  
  Punctuation should be processed as separate tokens.
* Considering the end task, obviously: keep stopwords.
* Lower-case or not?  
  An interesting possibility is *truecasing*: lower-case only the words at the beginning of a sentence, thus leaving acronyms (USA, CAT) and proper names (Marco Polo) with the correct starting letter.
* Numbers: keep all vs. replace all with <num> vs. keep only 1-digit and 4-digits   
  The last alternative appears to me as the most appropriate
* Non-latin words: if a word contains a character that is non-latin and not even in the Greek alphabet, such as “マギカ”, we replace the word with the <unk> token.

Example line from WikiText-2 (training set):

Father <unk> joined the observatory in 1889 after it was upgraded with more modern instruments . The same year , a second tower was built some 400 metres ( **1 @,@ 300** ft ) away from the main Gregorian Tower , overlooking the Vatican Gardens behind St. Peter 's Basilica on the **south @-@ west** border . It was built to a diameter of 17 metres ( 56 ft ) with a lower wall thickness of **4 @.@ 5 metres** ( 15 ft ) , which could bear the load of a 13 inch photographic <unk> , newly procured from Paris . ………’

Note: It is also necessary to reconvert some of the symbols found in the WikiText datasets, in particular:

‘@-@’ 4 ‘@.@’ 5 metres

Ampersand symbols taken away. Sentence-level tokenization:

'Father <unk> joined the observatory in 1889 after it was upgraded with more modern instruments .',

"The same year , a second tower was built some 400 metres ( **1,300** ft ) away from the main Gregorian Tower , overlooking the Vatican Gardens behind St. Peter 's Basilica on the **south-west** border .",

'It was built to a diameter of 17 metres ( 56 ft ) with a lower wall thickness of **4.5 metres** ( 15 ft ) , which could bear the load of a 13 inch photographic <unk> , newly procured from Paris .'

Without starting and ending spaces (the effect is seen elsewhere, 2 sentences before here), and lowercasing the first word in a sentence …

Tokens:

'father', '<', 'unk', '>', 'joined', 'the', 'observatory', 'in', '1889', 'after', …

Problem: I am decomposing the Unknown token, ‘<unk>’. …

Recap of the steps to process the text of a line in order to build a vocabulary that will be appropriate for a Language Model downstream task:

1. It is necessary to reconvert some of the symbols found in the WikiText datasets, in particular: ‘@-@’, ‘@.@’ etc.
2. nltk's tokenizer - sentence tokenizer
3. Eliminate outer spaces, and true-case the sentences.
4. Split into word tokens, and reunite the lls
5. Recreate <unk> from '<','unk','>'
6. There are word-tokens with non-Latin and non-Greek characters, such as: 匹, 枚, マギカ , etc. We replace those words with <unk>
7. Numbers. All decimals, together with all numbers that are not 1 digit (basic) or 4 digits (years), get replaced with <num>

Later on, these steps could be used to re-write the corpus and save it to disk for other uses.

The vocabulary should be created at the start of everything. Before retrieving the data from the KB sources.