Style transfer and classification in hebrew news items

Submitted as final project for the Deep Learning course, IDC, 2020

February 28, 2021

1 Introduction

Hebrew was classified as a Morphological Rich Language (MRL) by Tsarfaty et al. (2010)6. As such prefixes and suffixes are appended to words to change thier grammatical meaning. This structure has been shown by Tsarfaty et al (2019)6 to results in inherent morphological ambiguity in the language. For this reason, amongst others, Hebrew NLP algorithms lag behind other non MRL languages.

Recent developments in NLP, and new research in the field of Hebrew NLP bring promising prospects. In this work we implement, modify and examine two different Transformer based architectures for two Hebrew NLP tasks. We performed style transfer augmented text generation and classification of news items using a Vanilla Bert-like transformer, and a fine-tuned version of HeBert (Chirqui and Yahav, 2020)6. The learning in both cases was done over data scraped from *ynet* news portal's archive.

2 Related Works

Both the text generator and the classifier presented in this work are self-attention based transformers, using the same architecture described by Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser ans Polosukhin in their paper Attention is All you Need (2017)6. Transformers are a neural net architecutre designed to utilize *attention* (and *attention* only) to harvest context information in sequence transduction tasks. Older Recurrent neural net models utilized *cross-attention* to infer context in sequence-to-sequence transduction tasks. Recurent neural nets (RNN), Long short term memory neural nets (LSTM) and gated RNNs (GRUs) all generate a sequence of hidden states - h_t - as a function of the input for time t and the previous hidden state h_{t-1}. While this allows the neural net to learn and utilize context for each time t, context from far away positions is diminished as the same hidden state vector is overloaded with information. To solve this

problem a *cross-attention* mechanism was developed - *cross-attention* uses a weighted average of the different hidden states for each position, instead of one overloaded hidden vector. The weights of this weighted average are learned, allowing the network to decide when to give more 'attention' to different parts of the sequence, thus never overloading the hidden vector. Transformers introduced a new type of attention - *self attention* - which dicards recurrence all together in favor a new attention model. Transformers are built out of *attention heads* where each input x is embedded to some k dimensional vector and then undergoes three linear layers (seperatly) to produce three new vectors - Q ($Query$), K (Key) and V ($Value$). We proceed to compute a dot product of the $Query$ vector of each input x and the Key vectors of all other inputs in the sequence. This yields us a new vector A the length of the input sequence. A is than used as a weights vector to compute a context vector M in the same fashion as in *cross-attention*. Thus transformers can easly and quickly maintain context from any part of the sequence. Stacking many *attention-heads* yields more robut learning, and the architecture is well suited for transfer learning.

A very popular recent adaptation of transformers in NLP is BERT - Bidirectional Transformers for Language Understanding (Devlin, Chang, Lee, Toutanova 2018)6. BERT is used in both the classifier and partly in the text generator in this work. BERT is an architecture designed to allow pretraining of strong and robust language models over unlabeled text biderectionaly (ie conditioning on both left and right context in all layers). Such a pre trained BERT model can later be fine-tuned for a specific NLP task and a specific text corpus (of that same language). Fine tuning a BERT model is extermly quick and results in state of the art performacne. BERT pre-trained networks consists of many 'blocks' where each block consists of severl self-attention heads a residual connection, layer normalization and feed forward layers.

A recent major development in the field of Hebrew NLP was the introduction of HeBERT (Chirqui and Yahav, 2020)6. HeBert is a new and powerfull pre-trained BERT variant for Hebrew shown to outperform all other Hebrew NLP models on various language tasks such as Emotion detection, Fill-the-blank, NER, POS and sentiment analysis. Hebrew is considered a MRL language and as such needs more processing to disambiguate text. To solve this task the creators of HeBERT have tested various methods to tokenize Hebrew text in their paper. Different methods were empirically tested: char tokeniztion, different sub-words based tokenization, morpheme-based and finally word based tokenization. They've observed similar performance between between different sub-words based variants for unsupervised tasks, where char-based tokeniztion outperformed other methods in the Fill-in-the-blank task. For supervised tasks a 30K long sub-word tokenizer based on the YAP parser was used and shown to yield the best results.

YAP ('Yet Another Parser') - a joint morpho-syntactic parsing framework for processing Modern Hebrew texts (More and Tsarfaty, 2016)6. YAP is an extension of Zhang and Clarks's structure-prediction framework (2011)6. YAP recives a complete *Morphological Analysis* (MA) of some input text in the form of a *lattice* graph structure. The *lattice* structure contains all possible morphological analysis of the input text sequence, and each adjacent path in the graph represents a possible morphological segmentation of the sequence. Each morphological segmentation is used to build a dependency tree. Thus there are many dependencies trees (exponential to the number of possible paths). YAP proceeds to process the MA it two parallel dependent tasks: *Morphological Disambiguation* and *Dependency parsing:* The task of *Morphological Disambiguation* (MD)6 is to produce the most likely lattice-path out of a *Morphological Analysis* (MA). The task pf *Dependency parsing* (DEP)6 is to select the mpst likely dependency tree for a given path. For some text input x YAP jointly predicts a tuple $(MD(x), DEP(x))$. The predicted MD and DEP correspond with one another and are the analysis of the sentence with the highest probability.

Another variant of the original transformer model that was considered as a base for the classification task is the Multilingual Bert - mBert (Devlin, 2018)6. mBert was trained on 104 different languages from different families and has been shown by Wu and Dredze (2019) to have high performance as a zero-shot language transfer tasks. HeBert was shown to outperform mBert in classification tasks as shown in the HeBert paper.

3 Solution

3.1 General approach

3.1.1 Data collection and preprocessing

Data was collected from Ynet news portal's archive using a specialy built python web scraper. We've decided to scrape news data rather than use an existing Hebrew dataset as it featured a large amount of labeled data, including meta-data that could be used for style transfer. Such meta data such as tags appended for each article or the authors name and time of release would be later used for style transfer and were collected in later versions of the scraper. Furthermore, the news portals that feed our collective consiusness are an interasting corpus to study, as their analysis and tracing might reveal insights about our society and discourse.

3.1.2 Article title classification

A pre-trained HeBert6 sentiment analysis binary classifier was modified and fined tuned to classify article titles to the news section of which they came.

We've also considered the mBert6 net for this task, but decided to follow through with HeBert as it was shown to yield superior results. The only HeBert variant publically available is the binary sentiment analysis version which is built for a slightly different task, but since Bert and HeBert in particular is first trained on an independent corpus and than only fine tuned to a specific task we assumed the pre trained model behind the sentiment-analysis model will be a strong Hebrew language model, regardless of the task for which it was fine-tuned.

3.1.3 Text generation and style transfer

As HeBert uses a YAP based tokenizer we we're unable to fine tune it for text generation, as the generated tokens will not form a readable language. Moreover to include style transfer we would need an untrained network and preferably one that is easily modified. Thus, we've decided to train our own Bert-like encoder based text generator using char tokenization as it was shown in the HeBert paper to yield the best performence in unsupervised Hebrew tasks. Style transfer was achived by concating meta data about each char to it's embedded vector. Different ways to concat this data were tested.

3.2 Design

3.2.1 Data collection and preprocessing

We've collected over 1 GB of labeld data from the web archive of the news portal Ynet using a specially built python scraper that iterated over all articles in the archive since 1.1.2000. We chose to collect articles from 11 different news sections: (Military, Law and Justice, Health and Education, World Economy, Israeli Economy, General, Politics, Soccer, Palestinians, Sex and Sex and relationships) - many other news section were left out. Each article was collect into the follwing fields: Main title, Sub title, Article body, Label (news section), Author, Time of release, Tags added. For text generation each article was also processed in the following lines:

> "[SOS] main title [SEP1] sub title [SEP2] article_body [PAD] [EOS]"
>
> Each lines was truncated to 512 tokens, where the line was shorter than 512 padding was applied.

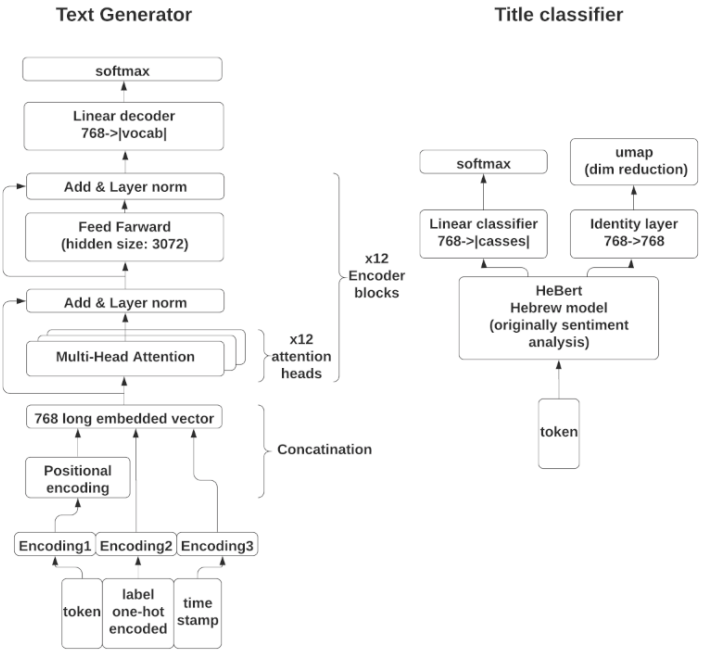

Figure 1: Network architecture

3.2.2 Article title classification

To adapt the HeBert sentiment analysis network for a multi-label classification task we've simply replaced the final linear layer with a new blank one of the right dimension and used CrossEntropy instead of BinaryCrossEntropy for a loss function. The original HeBert YAP based 30K long tokenizer was dowloaded from the transformers repository and used to tokenize the titles. A max length of 50 was chosen for every title. After fine-tuning the net we proceeded to perform dimensionality reduction on the output of it's final transformer layer (before the linear classifier). We will use this reduction to plot a 2D approximation of the representation of the classified items by our neural net, and gain further insight about it's inner workings.

3.2.3 Text generation and style transfer

The text generator net is built similarly to the Bert model and consists of an encoding layer, 12 layers of transformer encoder blocks and a final linear decoder layer activated by softmax. Two different approaches for embedding were tested: For each token it's 'news section' and 'time of release' data are processed into either a 10d long embedded vector using two linear layer, or a 2d long vector using simple min-max normalization.

News section classification	Article title (Hebrew)
General (כללי)	בת 60 נהרגה בתאונת דרכים בעוטף עזה
Law and Justice (משפט ופלילי)	אדם אבוטבול נעצר: גנב אופנוע במטרה לפגע
World economics (כלכלה עולמית)	האצה באינפלציה בגוש האירו בספטמבר
Military (צבא)	חיילים התבקשו לספר: "משהו מצחיק מצוק איתן"

Figure 2: Correct classifications

Correct classification	Predicted classification	Article title (Hebrew)
Israel economics (כלכלה ישראל)	General (כללי)	14% מתושבי ישראל מוותרים לעיתים על אוכל בגלל עוני
Military (צבא)	Palestinians (פלסטינים)	היום: שביתה כללית ברשות הפלסטינית
World economics (כלכלה עולמית)	General (כללי)	למי הועילה היעלמותו של סדאם חוסיין?
Health and education (בריאות וחינוך)	General (כללי)	כמויות מזוט גדולות זרמו לירקון

Figure 3: Incorrect classifications

The token itself is processed to either a 758 or 766 long vector corresponding. The token and style vectors were concatenated to a 768 long vector which was fed into the net. Each transformer block contains 12 self attention blocks, a layer norm and residual connection fllowed by a feed forward network. Finally a final soft maxed linear layer was used as a decoder. During training a square attention mask was used for each 512 long sequence. To generate text a start string, news section and time stamp are given to the generator, which recursivly feeds the trained model with longer and longer sequences until the model generates a [EOS] token or the 512 token limit is reached.

4 Experimental results

4.0.1 Article title classification

Data was scrambled and split 0.9-0.1 train and validation. The classification model reached an accuracy of 0.84 on validation set after two training Epochs using SGD and wAdam optimizer, each epoch taking less than 8 minutes on a Tesla p100-pcie-16gb GPU. Further training did not improve the results.

Examples of titles that have been correctly classificatied:

Examples of incorrect classifications:

Casting of each title's latent representation to a 2D array yielded the following scatter plot:

Further analysis of this graph and it's implementations on appendix A.

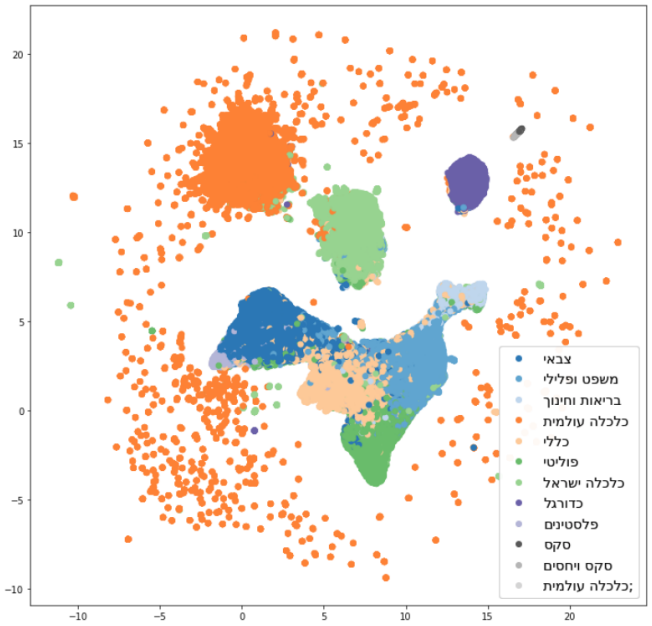

Figure 4: Scatter plot of scaled dim reduced latent representations

4.1 Article generator

Both models trained for 20 hours on a Tesla p100-pcie-16gb GPU exhibited very similar loss and preplexity metrics on the validation set. Learning did not stop after 20 hours and further improvment could be done using the same model. Text generation is an unsupervised task, making it's performance difficult to measure. We suffice with metrics on the validation set, qualitative observations and examples. The model seems to generate coherent sentances that relate to the starting vector and in most cases to the news section and time provided to it. Some outputs have minor grammatic flaws and some tend to recurce into a loop of the same few word. Most outputs are clear, logical, exhibit the required style and in some cases seem genuine.

GRAPH LOSS + PREPLEXITY

Examples of generated sentances:

GOOD EXAMPLES

BAD EXAMPLES

More examples given in appendix B

5 Discussion

Despite the ambiguity and morphological richness of Hebrew, it is evident that transformers in general and Bert in particular are a good tools for quick and transferable Hebrew models. Using the right tokenization and architecture and sufficient data Transformers can be used to generate compelling text and accurate classification. Furthermore we find that rich style transfer is easly achivable in transformers using simple concatination in the embedding layer. Assuming a concept c exists for the classification of all possible titles, assuming our trained model h is derived from an hypothesis space H that can predict c up to some ϵ bad hypothesis, and assuming sufficient amount of data we can derive that the latent representation of the items in the model are ϵ bad to thier representation in c. A model accurate up to 0.84 will have a latent representation of items that is accurate to the same degree. Thus an accurate 2D representation of items yields many insights on how the data is not only represented in the network, but also to as how it is represented in c up to ϵ. Dense clusters are news sections that are more focused in content, sparse clusters come from news sections that cover many varying topics. Close clusters are of news sections that cover closely related possibly covering the same topics or having the same writers, far away clusters the opposite. Furthemore any text can be ran through the net and casted on the scatter plot - assuming all of the above, it's location will reveal it's place in the discourse of the corpus that has been learned. Examples of such castings are shown in appendix A.

Further work can be done in these two networks: scraping more data from more news sections and possibly more news portals, allowing the text generator more time to learn, learning style for author and tags. Char vocabulary should be pruned to allow quicker learning. Finally the pretrained generator should be tested as a pre-trained Hebrew model for various fine tuning NLP taks.

TAMIR? ELABORATE ON UMAP FUNCTION AND ACCURACY?

6 Code

Link to the github repository with the notebooks: `https://github.com/hopl1t/ynet_nlp.git`

Link to data pickle file at Google drive:

`https://drive.google.com/file/d/1fyRLecpSduPzSeQhgOPu-puROAqqgVf1/view?usp=sharing`

Link to the first trained text generator pickle file at google drive (style transfer 2d):
`https://drive.google.com/file/d/1Q7rsobin53tkF7lZICi2S5yeJsYcBe5u/view?usp=`

sharing

Link to the second trained text generator pickle file at google drive (style transfer 10d): https://drive.google.com/file/d/1MbEpF_1r0nXQQFO9DqeD8L7XEwP2hz-c/view?usp=sharing

Link to the second trained classifier pickle file at google drive (style transfer 10d): https://drive.google.com/file/d/11Q3AgI-iIitfrRqmvk9sh7Xjjiwe6CNj/view?usp=sharing

References

Tsarfaty R, Seddah D, Goldberg Y, K¨ubler S, Versley Y, Candito M, Foster J, Rehbein I, Tounsi L (2010) Statistical parsing of morphologically rich languages (spmrl) what, how and whither. Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages, 1–12.

Jacob Devlin. 2018. Multilingual bert readme document

Reut Tsarfaty, Amit Seker, Shoval Sadde, Stav Klein. 2019. What is wrong with Hebrew NLP? And how to make it right

Amir More and Reut Tsarfaty. 2016. Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. In Proceedings of COLING, pages 337–348. The COLING 2016 Organizing Committee.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. Computational Linguistics, 37(1):105–151.

Avihay Chirqui and Inbal Yahav. 2020. HeBERT & HebEMO: a Hebrew BERT model and a tool for polarity analysis and emotion recognition

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. Computational Linguistics, 37(1):105–151.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In Proceedings of the ACL, HLT '11, pages 188–193, Stroudsburg, PA, USA. ACL.

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin. 2017. Attention is All you Need.

Devlin, Chang, Lee, Toutanova. 2018. BERT: Pre-training of Deep Bidirectional transformers for Language Understanding.

7 Appendix A - more scatter plots

Stuff on scatter

8 Appendix B - more generated text

GOOD EXAMPLES

BAD EXAMPLES