# Seminar Project: Sentence Similarity Models

Students: Sara Bonati - Irina Kokoshko
Course: Natural Language Processing (Instructor: Prof. Dr. Arthur Jacobs)
Programme: MSc Data Science (Freie Universitat Berlin)

## Contents



## 1 Repository Organization

Project scripts, in form of Jupyter notebooks, and data files can be found in the project's Github repository. More specifically, the repository is organized as follows:

- **Data** folder: this folder contains the sentence data files for the three different datasets tested in the project. The files are either in .csv, .txt or pickle file form and are handled in the scripts already using the appropriate reading functions. The files are:
  - `sentences_shorter.pkl` is a pickle file containing the sentences that we call as STS1 dataset. The data is organized as a pandas dataframe with 3 columns (sentence1, sentence2 and gold standard score for the sentence pair). The sentences and gold standard scores come from .txt files `STS.input.MSRpar.txt`,`STS.gs.MSRpar.txt`,`STS.input.MSRvid.txt` and `STS.gs.MSRvid.txt` respectively. These txt files are also found in the Data folder.
  - `sts.txt` is a text file containing the sentences that we call as STSFull dataset (comprising sentences from multiple STS benchmark editions and years). This file is read in the scripts in its original form (text file): during the scripts' execution the data is organized as a pandas dataframe with columns specifying the data and scores (sentence1, sentence2 and gold standard score for the sentence pair) and other columns specifying

e.g. year, sentence pair category etc. For the purposes of this project we focus solely on the (sentence1, sentence2 and gold standard score for the sentence pair) columns.

  - `data_sent_exp_2021-02-20_16-31.csv` is a .csv file containing the survey results as provided by SoSciSurvey.
  - `dog bites man vs man bites dog.xlsx` is an Excel file containing the stimuli (sentence pairs) used in the online SoSciSurvey questionnaire.
  - `survey.pkl` is a pickle file containing the survey sentences, along with their respective "gold standard" scores obtained by survey participants. The data is saved as a pandas dataframe and is obtained by preprocessing the files `dog bites man vs man bites dog.xlsx` and `data_sent_exp_2021-02-20_16-31.csv`. The data is organized as a pandas dataframe with 3 columns (sentence1, sentence2 and gold standard score for the sentence pair).

- **Scripts** folder: this folder contains the scripts which, for each model tested, load the data, apply the model and calculate the correlation between gold standard scores of each dataset and NLP model predictions for each dataset. The files are divided by model to avoid long scripts, more specifically we have:

  - `apply_model_Li.ipynb` is a Google Colab/ Jupyter notebook which applies the Li model to the three datasets we test
  - `apply_model_BERT.ipynb` is a Google Colab/ Jupyter notebook which applies the BERT model to the three datasets we test
  - `apply_model_SBERT.ipynb` is a Google Colab/ Jupyter notebook which applies the SentenceBERT models to the three datasets we test
  - `Survey_prepare_data.ipynb` is a Google Colab/ Jupyter notebook which takes as input the survey sentences Excel file and SoSciSurvey Excel results file and creates a pickle file which can be readily imported in all the above model scripts to test NLP models on survey data. It is not necessary to run this file to test the code, as the pickle file is already prepared, but here we included it in the Scripts folder to show all steps of data preparation.
  - `prepare_train_data.py` is a Python script that creates the `sentences_shorter.pkl` pickle file from the original .txt files containing the sentences (files in the Data folder). It is not necessary to run this file to test the code, as the pickle file is already prepared, but here we included it in the Scripts folder to show all steps of data preparation.

- **Figures** folder: this folder contains two scripts which create figures for the Li model and BERT/SBERT models respectively. We have put these two files in a separate folder since some of the analyses for these models can take a long time to be completed, hence we put the results already in these scripts to create figures without the need to run all the NLP models' scripts. The files are:

  - `gridsearch_figures.ipynb` is a Google Colab/ Jupyter notebook that creates Figures 1 and 2, which can be seen in Section 7.
  - `BERT_figures.ipynb` is a Google Colab/ Jupyter notebook that creates Figures 3 and 4, which can be seen in Section 7.

# 2 Methods

## 2.1 Datasets

There are three datasets used in this seminar project. The first two are different versions of the STS benchmark. The STS benchmark comprises sentence pairs from different sources (such as news outlets or internet forums) along with a semantic similarity score $s$. The first dataset

we test is a subset of the STS benchmark used in the 2012 competition, and it includes (after preprocessing) $n = 1218$ sentence pairs taken prevalently from news outlets: in the scripts we refer to this dataset with the name **STS1**. The second dataset we tested is aggregated sentence pairs STS data from multiple sources for years between 2012 and 2017. This second version includes more data sources like internet forums or captions and has a total of $n = 5506$ sentence pairs (after preprocessing): in the scripts we refer to this dataset with the name **STSFull**. For STS1 and STSFull a semantic similarity score $s \in [0, 5]$ is also given for each sentence pair. Lastly, we also use a dataset of semantically ambiguous $n = 100$ sentence pairs created by us: in the scripts we refer to this dataset with the name **Survey**. In creating the dataset we tried to use phrases which have no lexical or sublexical mistakes but whose semantic information can be confusing or "nonsense" in some cases. Since this dataset was created ad hoc there are no semantic similarity scores associated with the sentence pairs, therefore a short survey was conducted to collect human ratings.

## 2.2 Survey

For the semantically ambivalent dataset we collected ratings from human participants in an online survey conducted on the SoSciSurvey platform (link to the survey: https://www.soscisurvey.de/sent_exp/).

**Participants and Stimuli** A total of 31 participants took part in the SoSciSurvey questionnaire. We excluded participants who did not fully complete the survey, retaining a total of 21 participants for the final analysis. A total of $n = 100$ sentence pairs were written for the survey. Sentences were varied to include "nonsense" themes as well as simple lexical manipulations (e.g. "Romeo loves Juliet" VS "Juliet loves Romeo") to obtain a variety of semantic ambiguity problems.

**Procedure** First, participants received detailed written instructions about the sentence semantic similarity task: participants were instructed to indicate the perceived semantic similarity score between two sentences with a percentage comprised between 0% (no semantic similarity at all) and 100% (full semantic similarity). Before completing the real survey participants were provided with some examples to familiarize themselves with the online response process; they were also requested to fill out a series of questionnaires measuring behavioural variables (nationality, age, gender, level of English proficiency, frequency of reading in a week). Finally, participants completed one experimental run of the survey with the sentence pairs from the dataset, organized in 5 questionnaire pages with 20 sentence pairs each. The order of sentence pairs presentation was randomized: there was no maximum trial duration (so participants were free to take their time to answer) but participants were obliged to give an answer (no blank responses allowed).

## 2.3 Preprocessing

**Regular expressions and data cleaning** Lexical elements like punctuation or symbols can be tricky to interpret for NLP models and can distort the final results. We prepared our data by applying some regular expression searches and simplifying instances of search matches. Contrary to prevalent preprocessing pipelines we only transform the text in lowercase and perform tokenization: this means that there is **no** stopwords exclusion or lemmatization executed. This choice is motivated by the fact that, while for bigger semantic tasks like document similarity stopwords form redundant and abundant information, for sentences (especially short ones where a few words can convey great meaning) these elements can still convey a lot of information and give a different meaning/connotation to a phrase. Some research also highlights the importance of stopwords and leaving the text "as it is" as much as possible to avoid losing precious information. The regular expression queries we run include symbol removal, extending contractions and more, inspired by some of the queries we saw online applied to STS benchmarks. Additionally, in creating the STS1 dataset we restricted the sentence pairs used to sentences with a maximum of 25 words (to avoid long and complex sentences).

**Semantic similarity score normalization.** The original datasets provide semantic similarity scores whose range is $[0, 5]$. Since most models use cosine similarity as a semantic similarity score, whose range is comprised between 0 and 1, we normalize the scores of the datasets STS1 and STSFull to get a semantic similarity score $s \in [0, 1]$. For the Survey data this process is not necessary since participants rated the semantic similarity of sentence pairs with a percentage between 0% and 100%, translating directly to range $[0, 1]$.

# 3 Model formulation

## 3.1 Li model: fully semantic information

The first model we test is a semantic graph-based model from [Li et al., 2006]. The premise of the paper is to study sentence similarity problems, and to formulate a model that takes into account the peculiarities of sentences. Sentences are longer than single words and include more semantic information as more words are added to a sentence, but at the same time it can be hard to discern the semantic context of a sentence from the few words composing it, especially when compared to big documents with thousands of words. The authors also note that while many words such as stopwords or conjunctions can be removed from big documents (as they only introduce unnecessary semantic "noise") often these words can have a big impact on the semantic meaning of a sentence. The Li model for sentence similarity is specifically targeted to shorter sentences and its steps can be summarized as follows:

1. Creation of a common word set $T$ containing all unique words used in sentence 1 and sentence 2
2. Creation of semantic vectors $s_1$ and $s_2$ for both sentences
3. Creation of word order vectors $o_1$ and $o_2$ for both sentences
4. Evaluation of Li semantic similarity score as a mixture of similarity between semantic and word order vectors

The model calculations are controlled by 5 parameters $\{\alpha, \beta, \eta, \phi, \delta\}$. The Li model parameter set is composed of:

- $\alpha \in [0, 1]$ is a parameter used to calculate the path length between two words in a semantic net hierarchy
- $\beta \in (0, 1]$ is a smoothing factor taking into account the depth of word in the hierarchy
- $\eta \in [0, 1]$ is a word order threshold
- $\phi \in [0, 1]$ is a semantic threshold
- $\delta \in (0.5, 1]$ is a parameter defining the relative contributions of semantic and word order information to the overall similarity computation.

The implementation of the above mentioned steps operates with two information sources that are used in the creation of the semantic and word order vectors. WordNet is used as a semantic knowledge base for the calculation of semantic similarity, while the Brown Corpus is used to provide the necessary statistical information, namely the probability of a word w in the corpus.

## 3.2 State of the art vector space models: BERT and SBERT

The advent of deep learning and deep model architecture has greatly influenced NLP and led to the development of powerful NLP models [Deng and Liu, 2018, Liu et al., 2020]. Among these we find GPT, GPT2, and BERT. In this project we focus on BERT (Bidirectional Encoder Representations from Transformers), a state of the art NLP language model that can be applied to a

variety of language tasks including sentence similarity [Devlin et al., 2018]. Based on the popular Transformer achitecture [Vaswani et al., 2017], what distinguished BERT from other models is the ability to create high-dimensional word embeddings using contextual information coming both from words before the current word and after the current word (bidirectional self-attention). This allows BERT to build a word embedding that takes into account the context in which a particular word is used, a feature that proves crucial in e.g. word meaning discrimination. The BERT model usually works on the word level first, creating a word embedding for each word in a sentence, and then aggregates the word embeddings together to form a sentence embedding/vector. In the project we use a pretrained BERT model from the `transformers` Python package. More specifically, we use the **bert-base-uncased** model version. Details on the training data this model was trained on, as well as other hyperparameters like training batch size and learning rate, can be found at https://huggingface.co/bert-base-uncased. In the current project we were interested in testing different variations of how BERT treats word embeddings to form a final sentence embedding, and how these variations can impact the final correlation with human ratings. We experiment with the following settings:

- *Number of hidden layers used to form a single word embedding*: for each word in a sentence we try to form its respective word embedding by using $n$ BERT hidden layers, where $n \in [1, 12]$ (1 represents just the last hidden layer, while 12 represents all the hidden layers of the BERT architecture excluding the initial CLS token hidden layer).
- *Aggregation strategy used to merge together all hidden layers into a word embedding*: for each word in a sentence, once the respective hidden layers are selected we try to aggregate the hidden layers together either by averaging them / concatenating them / summing them to form a word emebdding.

Once all word embeddings are formed the sentence embedding is formed in the most popular way, namely by averaging all the word embeddings together. Similarity between the two sentence embeddings $s_1$ and $s_2$ is then calculated with cosine similarity $c(s_1, s_2)$:

$$c(s_1, s_2) = \frac{s_1 s_2}{||s_1||\,||s_2||}$$

We also try a variation of BERT called SentenceBERT [Reimers and Gurevych, 2019], which can be used as a Python package to get a sentence embedding vector directly rather than having to write code focusing on word embedding operations first. The SentenceBERT package offers a variety of pretrained models differing in the amount of parameters, the pooling strategy, and the data used for training and in some cases fine tuning. The pre-trained models used in this project are (source: https://docs.google.com/spreadsheets/d/14QplCdTCDwEmTqrn1LH4yrbKvdogK4oQvYO1K1aPR5M/edit#gid=0):

| **SBERT model** | **Base model** | **Pooling strategy** | **Training data** |
|---|---|---|---|
| sts-bert-base | bert-base-uncased | Mean | NLI + STSb |
| sts-bert-large | bert-large-uncased | Mean | NLI + STSb |
| sts-roberta-large | roberta-large | Mean | NLI + STSb |
| nli-bert-base-cls | bert-base-uncased | CLS token | NLI |
| nli-bert-base-max | bert-base-uncased | Max | NLI |
| paraphrase-distilroberta-base | distilroberta-base | Mean | Paraphrase data |

# 4 Results

## 4.1 Li model

We first look at the results of the Li model with the original parameter set used in [Li et al., 2006]. The recommended parameter values are $\alpha = 0.2, \beta = 0.45, \phi = 0.2, \eta = 0.4, \delta = 0.85$. For this

model we visualize Pearson correlation between human ratings and model predictions for the three datasets in the table below.

| **Li Model (using recommended parameters from** [Li et al., 2006]) | |
|---|---|
| Dataset | Correlation |
| STS1 | 0.229 |
| STS Full | 0.569 |
| Survey | 0.092 |

The Li model performs best on the STSFull dataset achieving a correlation value of 0.57. Next, we execute a grid search over parameter space to see if different parameter combinations can improve the performance. Due to the dataset size and the time needed to perform the grid search we report results for the STS1 dataset only. Figure 1 shows the grid search over $(\alpha, \beta)$, where $\alpha \in [0, 1]$ and $\beta \in (0, 1]$; Figure 2 shows the grid search over $(\eta, \phi)$, where $\eta \in [0, 1]$ and $\phi \in [0, 1]$. Indeed, changing the parameters increases the correlation score, but only by a few percent, from .23 to .26. The best found parameter values from the two grid searches are $\alpha = 0.8, \beta = 0.4$ and $\phi = 0.6, \eta = 0.9$, respectively.

## 4.2 BERT and SBERT

First, we report the results of the BERT Base model from the `transformers` package. As previously explained we vary the BERT model application by changing the number of hidden layers $n_l$ used to form a word embedding $n_l \in [1, 12]$ and by changing the pooling method used to create the word embedding from the hidden layers (average, sum and concatenation respectively).

| BERT Model | | | | |
|---|---|---|---|---|
| Dataset | N. Layers used | Correlation (Average) | Correlation (Concat) | Correlation (Sum) |
| STS1 | 1 | 0.200 | 0.200 | 0.200 |
| | 2 | 0.208 | 0.206 | 0.208 |
| | 3 | 0.209 | 0.207 | 0.209 |
| | 4 | 0.214 | 0.211 | 0.214 |
| | 5 | 0.216 | 0.213 | 0.216 |
| | 6 | 0.216 | 0.214 | 0.216 |
| | 7 | 0.215 | 0.214 | 0.215 |
| | 8 | 0.217 | 0.217 | 0.217 |
| | 9 | 0.219 | 0.220 | 0.219 |
| | 10 | 0.222 | 0.223 | 0.222 |
| | 11 | 0.224 | 0.226 | 0.224 |
| | 12 | 0.227 | 0.229 | 0.227 |
| STS Full | 1 | 0.546 | 0.546 | 0.546 |
| | 2 | 0.556 | 0.552 | 0.556 |
| | 3 | 0.553 | 0.547 | 0.553 |
| | 4 | 0.551 | 0.543 | 0.551 |
| | 5 | 0.558 | 0.547 | 0.558 |
| | 6 | 0.564 | 0.551 | 0.564 |
| | 7 | 0.570 | 0.555 | 0.570 |
| | 8 | 0.576 | 0.560 | 0.576 |
| | 9 | 0.583 | 0.565 | 0.583 |
| | 10 | 0.591 | 0.572 | 0.591 |
| | 11 | 0.601 | 0.581 | 0.601 |
| | 12 | 0.611 | 0.591 | 0.611 |
| Survey | 1 | 0.301 | 0.301 | 0.301 |
| | 2 | 0.304 | 0.309 | 0.304 |
| | 3 | 0.302 | 0.306 | 0.302 |
| | 4 | 0.297 | 0.303 | 0.297 |
| | 5 | 0.286 | 0.297 | 0.286 |
| | 6 | 0.270 | 0.289 | 0.270 |
| | 7 | 0.256 | 0.281 | 0.256 |
| | 8 | 0.240 | 0.271 | 0.240 |
| | 9 | 0.223 | 0.261 | 0.223 |
| | 10 | 0.207 | 0.251 | 0.207 |
| | 11 | 0.191 | 0.240 | 0.191 |
| | 12 | 0.177 | 0.231 | 0.177 |

Overall we can see that in the case of STS1 we reach a minimum correlation score of 0.2 (1 hidden layer used, same score across all aggregation strategies) and a maximum correlation score of 0.229 (12 hidden layers used, with concatenation as aggregation strategy). We observe that a higher number of hidden layers used corresponds to a higher correlation score, and no significant difference between the aggregation strategies can be observed. For STSFull we see a similar trend (higher number of hidden layers used corresponds to a higher correlation score, no significant difference between the aggregation strategies can be observed): minimum correlation score of 0.545 (1 hidden layer used,same score across all aggregation strategies) and a maximum correlation score of 0.611 (12 hidden layers used, with concatenation as aggregation strategy). Interestingly for the Survey dataset we observe an opposite trend: this time a higher number of hidden layers used leads to a worse correlation score, with no significant difference between the aggregation strategies (although concatenation seems to hold better than average or sum the more hidden layers get used). The survey results also show this trend: minimum correlation score of 0.177 (12 hidden layers used, average as aggregation strategies) and a maximum correlation score of 0.308 (2 hidden layers used, with concatenation as aggregation strategy).

Now we turn to the results of SentenceBERT models. We test six different models varying in the number of parameters (e.g. bert base model compared with the larger model bert-large), type of pooling used to create sentence embedding vector, and finally differing also in the training and fine-tuning data used.

| **SentenceBERT Model** | | |
|---|---|---|
| SentenceBERT model name | Dataset | Correlation |
| sts-bert-base | STS1<br>STSFull<br>Survey | 0.149<br>0.986<br>0.091 |
| sts-bert-large | STS1<br>STSFull<br>Survey | 0.151<br>0.989<br>0.050 |
| sts-roberta-large | STS1<br>STSFull<br>Survey | 0.145<br>0.979<br>0.216 |
| nli-bert-base-cls | STS1<br>STSFull<br>Survey | 0.130<br>0.763<br>0.070 |
| nli-bert-base-max | STS1<br>STSFull<br>Survey | 0.144<br>0.741<br>0.052 |
| paraphrase-distilroberta-base | STS1<br>STSFull<br>Survey | 0.169<br>0.828<br>0.310 |

Overall, we can see that the SentenceBERT models also perform very well. In particular, it was expected to reach a very high correlation score in the STSFull dataset for the first three pretrained models, since these models have been trained on this very same data. However even with different training data the other models still manage to get a good correlation $\approx 0.7/0.8$. In STS1 the sentenceBERT models all have a similar performance but slightly smaller correlation values than other models, reaching correlations of $\approx 0.13$ to $\approx 0.17$. Interestingly, for the Survey dataset we see the best performance in one of the sentenceBERT models: the paraphrase model reaches a correlation with gold standard scores from human raters of 0.31, the highest among all models tested for the difficult and ambiguous survey sentence pairs.

# 5 Discussion

**Limitations.** The semantic similarity project has some limitations, which we present divided in three main categories: data and survey, preprocessing and models.

- **Data and survey.** We chose versions of the STS benchmark as the data to use in our project because it provided gold standard scores (obtained by averaging ratings of multiple human raters) for each sentence. This saved us recruiting a lot of participants, but it also constrained us to use the sentences in the STS benchmark, a few of which come from very specific semantic domains (e.g. finance or politics) and present a lot of subordinates and complex syntactic constructions. For semantically ambivalent phrases no gold standard data was provided, so we decided to perform a survey on our own. We note here that the survey was not pre-tested and improvements on the type of stimuli/sentences and the number of participants ($n = 21$) could have been made.

- **Preprocessing.** An integral part of every NLP pipeline is preprocessing. The task of clearing language data is especially tricky for sentences: in some cases elements such as punctuation

or numbers, which would normally be discarded, can really shape a sentence's meaning e.g. "I got a 1.0 in the test" VS "I got a 5.0 in the test". In our preprocessing pipeline we used regular expression queries to remove elements such as symbols but decided to keep other elements such as numbers or symbols like "!". This decision was motivated by the type of sentences we had in our dataset e.g. news from finance, whose meaning heavily relies on number elements in the sentences, however we do recognize that keeping or removing some sentence elements may have negatively influenced the models' performance.

- **Models.** While the Li model has access to semantic graph information provided by the WordNet API, in the original paper formulation no strategies for word sense/meaning disambiguation are implemented. This is a critical point for sentence similarity, as the Li model would not distinguish e.g. the word sense of "can" in "I can do this" VS in "I open a can of tomatoes". In contrast, BERT and SentenceBert can distinguish between different meanings of a single word, thanks to the bilateral attention mechanisms and the plethora and variety of data these models were trained on. If there is a limitation to be noted for the Transformer models it could be interpretability, as the "dark-box" weight matrices and complex network architectures do not provide the same clarity that comes with semantic graphs. In addition, we tested these models using pre-trained architectures due to hardware constraints on our local machine. Should one decide to train these models from scratch on new training data a lot of GPU/computing power would be required to speed up the training and fine-tuning processes.

**Conclusions.** In this seminar project we have tested different models to test semantic similarity score predictions of NLP systems against human ratings. Model assumptions, as well as model architectures and sentence embedding dimensionality, prove crucial to reach a good correlation with human performance. As expected the recent state of the art models like BERT, or variations of BERT as explored with the SentenceBERT pretrained models, often surpass simpler or older models in terms of correlation with gold standard scores from human raters. However results must be interpreted with caution: as highlighted in the limitations, factors like preprocessing queries or simply the nature of the sentence data can have a big impact on the final performance regardless of the model architecture and complexity (e.g. the poor performance of an advanced model like BERT on the survey data). Semantic ambiguity as represented in the survey also remains a tricky task for all models. Overall we believe that future models should implement both advanced methods like bidirectional self-attention (see BERT) but also try to implement efficient strategies to highlight semantic and syntactic information: this could be achieved for example by working closely with a POS tagger or by exploring in depth the semantic tree of a word, with a final goal of relating these explorations to vector space models (VSMs). A step in this direction can be already seen in some papers like [Pittaras et al., 2020]. While these new proposed approaches have focused on text classification so far, the idea of merging semantic graphs and VSMs is an interesting proposition which could help shape better and more powerful models for sentence similarity in the future.

# 6 Acknowledgements

We would like to thank Prof. Dr. Arthur Jacobs for providing us with Li model code and useful advice on the project. Finally, we also thank all the participants of the sentence semantic similarity survey. The following online resources were used:

- https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/
- https://github.com/TharinduDR/Siamese-Recurrent-Architectures/tree/master/preprocessing
- https://github.com/UKPLab/sentence-transformers

# References

[Deng and Liu, 2018] Deng, L. and Liu, Y. (2018). *Deep learning in natural language processing.* Springer.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

[Li et al., 2006] Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D., and Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering*, 18(8):1138–1150.

[Liu et al., 2020] Liu, Z., Lin, Y., and Sun, M. (2020). *Representation Learning for Natural Language Processing.* Springer Nature.

[Pittaras et al., 2020] Pittaras, N., Giannakopoulos, G., Papadakis, G., and Karkaletsis, V. (2020). Text classification with semantically enriched word embeddings. *Natural Language Engineering*, pages 1–35.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762.*

# 7 Figures

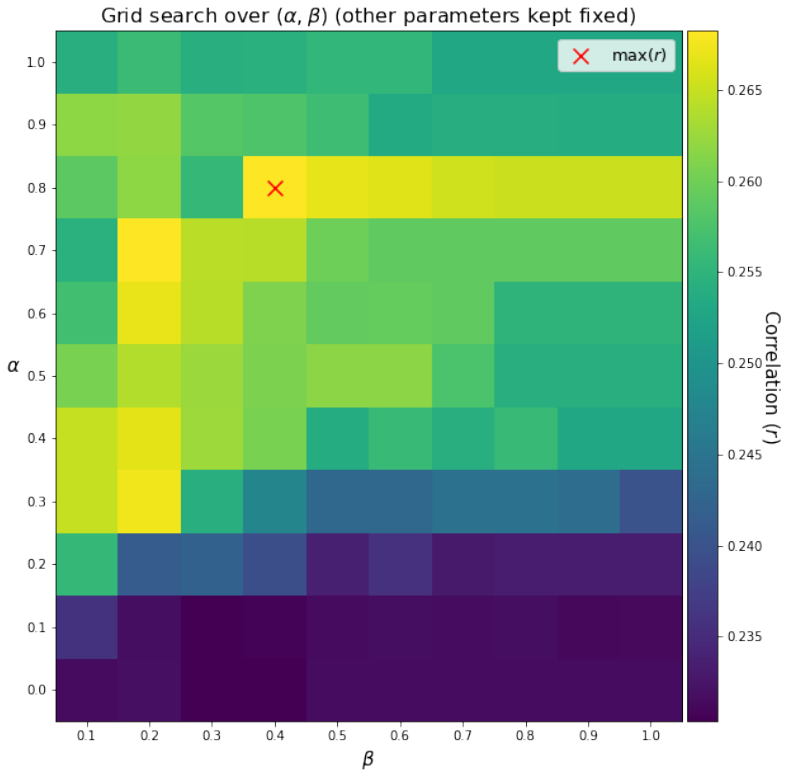


Figure 1: Grid search over the $(\alpha, \beta)$ parameter space, keeping other parameters fixed at values recommended in [Li et al., 2006].

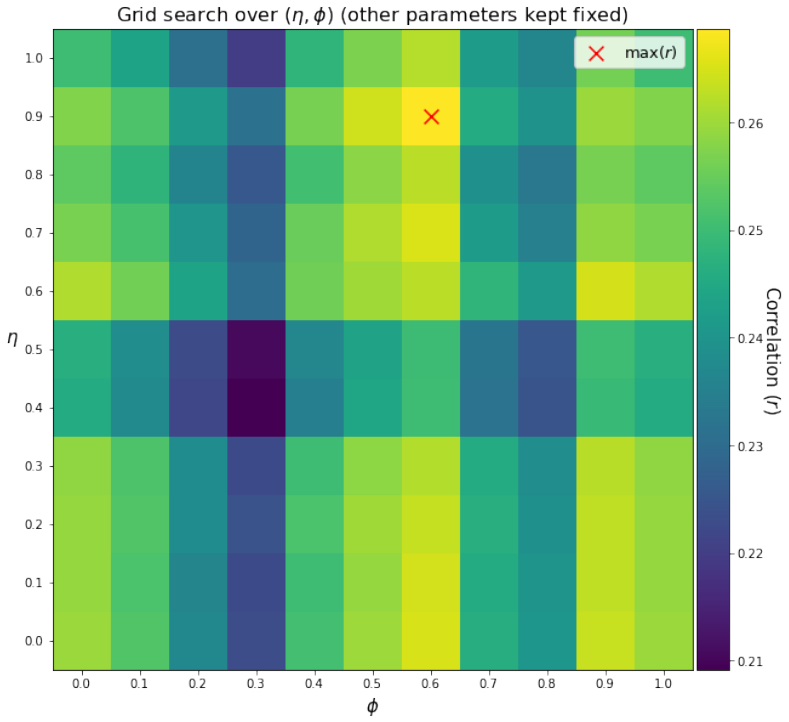


Figure 2: Grid search over the $(\eta, \phi)$ parameter space, keeping other parameters fixed at values recommended in [Li et al., 2006].

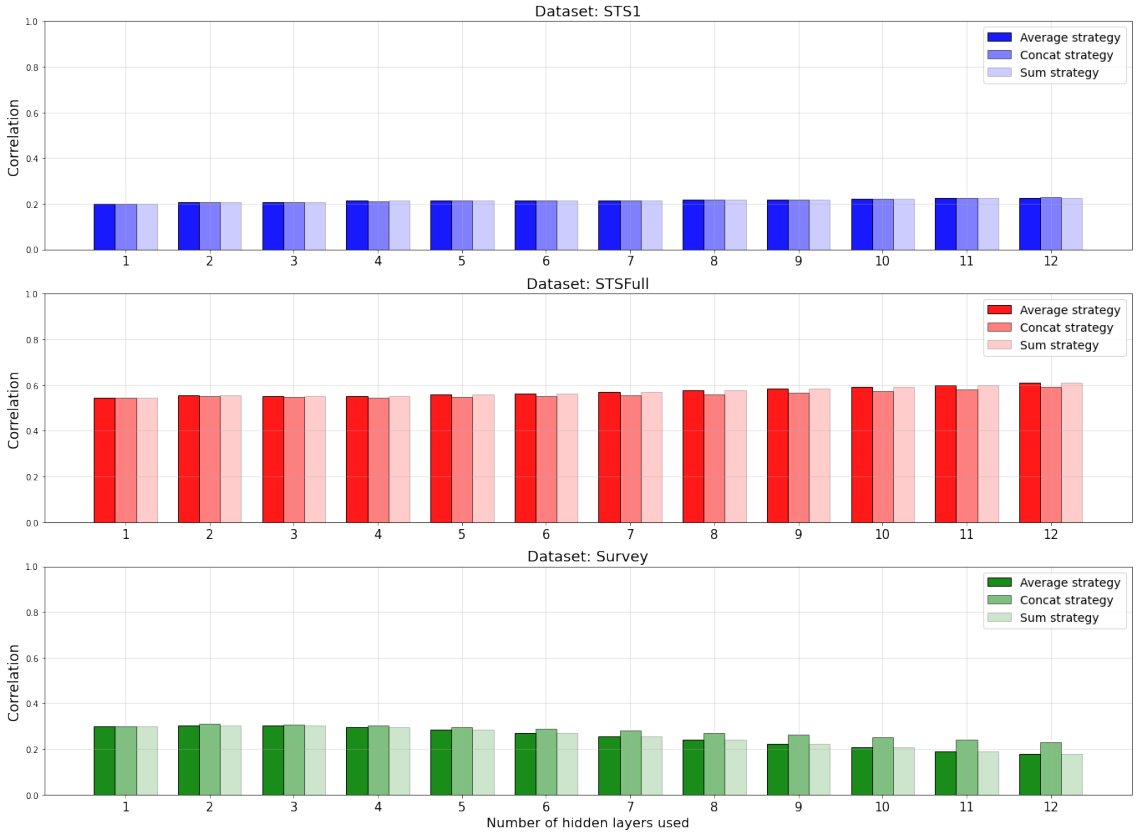


Figure 3: **BERT results.** The Figure shows the correlation between BERT models and gold standard scores for each dataset. The BERT models differ in aggregation strategy used (color coded for each dataset as displayed in the legend) and in number of hidden layers used (as displayed in the x axis of the plots). For STS1 and STSFull the overall trend seems to be an increase in correlation score as the number of hidden layers used increases; for Survey data we observe a decrease in correlation score as the number of hidden layers used increases.

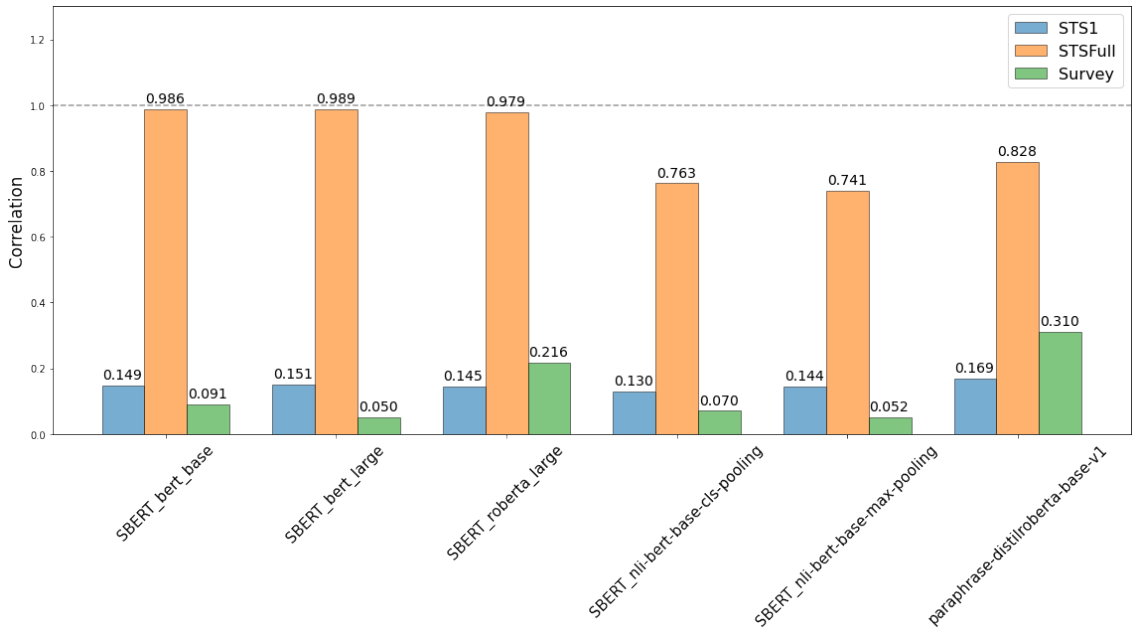


Figure 4: **SBERT results.** The Figure shows the correlation between SBERT models similarity scores (each model is displayed in the x axis) and gold standard scores for each dataset (color coded as displayed in the legend).