SimBa at CheckThat! 2022: Lexical and Semantic Similarity Based Detection of Verified Claims in an Unsupervised and Supervised Way

Alica Hövelmeyer[1,2], Katarina Boland[1,2] and Stefan Dietze[1,2]

[1]Heinrich-Heine-Universität Düsseldorf (HHU), Universitätsstraße 1, 40225 Düsseldorf, Germany
[2]GESIS - Leibniz Institute for the Social Sciences, Unter Sachsenhausen 6-8, 50667 Cologne, Germany

Abstract

One step in many automated fact-checking pipelines is verified claim retrieval, i.e. checking whether a claim has been fact-checked before. We approach this task as a semantic textual similarity problem. For this, we examine the extent to which an input claim and a verified claim are similar at semantic, textual, lexical and referential levels using a variety of NLP tools. We rank similar pairs based on these features using a supervised and an unsupervised model. We participate in two subtasks and compare our results for subtask 2A: *detecting previously fact-checked claims from tweets* and subtask 2B: *detecting previously fact-checked claims in political debates* for English data. We find that the combination of semantic and lexical similarity features performs best in finding relevant claim pairs for both subtasks. Furthermore, our unsupervised method is on par with the supervised one and seems to generalize well over similar tasks.

Keywords

fact-checking, STS, semantic similarity, lexical similarity, sentence embeddings

1. Introduction

The dissemination of true or false information through traditional channels, such as political speeches, or channels that have emerged in recent years, such as social media, is a powerful tool for shaping public opinion. Therefore, the analysis of claims made online or by public speakers is a popular field of research. The *CLEF CheckThat Lab*[1] contributes by offering shared tasks related to it. This paper reports on our submission for Task 2: Detecting previously fact-checked claims for English language.

We approach this task as a semantic textual similarity problem that we solve by combining different kinds of similarity features. We want to build on the success of sentence embedding models by trying out different models, their combinations and different ways of weighing them. In addition, we also want to contribute to a better understanding of sentence embeddings by investigating what kinds of possible similarities between sentences they capture and how their performance can be improved by adding complementary information. The combination of lexical and semantic similarity features proves to be particularly helpful. The great strength of

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
alica.hoevelmeyer@hhu.de (A. Hövelmeyer); katarina.boland@hhu.de (K. Boland); stefan.dietze@hhu.de (S. Dietze)

our approach is that we compare a supervised and an unsupervised method to rank the given data by similarity and are able to propose an unsupervised method that is on par with supervised approaches. Furthermore, there is evidence that our unsupervised method generalizes well over similar tasks. The code for both subtasks is available on github[1].

2. Related Work

This submission is part of the 5th edition of the CheckThat! Lab. Previous editions, also held in conjunction with the Conference and Labs of the Evaluation Forum (CLEF) that also featured the task of Detecting Previously Fact-Checked Claims / Claim Retrieval, took place in 2020[2] and 2021[3]. The approaches proposed by the participants are similar to ours in various aspects.

For the lab in 2020 the data to be processed exclusively consisted of tweets as input claims. Many of the participants used pre-processing and cleaned the tweets, removing tweet-specific characters like hashtags [4] [5] [6] [7]. Some teams solely made use of lexical and string similarity features[5] [6], whereas other teams used pre-trained language models to evaluate semantic similarity. These teams fine-tuned *RoBERTa*[8][4] or used *Sentence-BERT*[9] [10] [11] [7] or *Universal Sentence Encoder*[12][13] in order to calculate the distances between sentence embeddings. Different variations of *Blocking*-techniques were also used[10][5] [7]. Similar to our approach, some teams combined lexical and semantic similarity features [4] [13] [11].

In 2021 all teams made use of the sentence embedding model Sentence-BERT. Team NLytics[14] offered an unsupervised approach based on the distances of sentence embeddings gained using Sentence-BERT. This approach performed well for only one of the proposed subtasks.

Team DIPS[15] and Team Aschern[16] made use of the combination of a semantic similarity feature (also gained using the sentence embedding model Sentence-BERT) and a string (*BM25* by Team DIPS) or lexical (*TF.IDF* by Team Aschern) similarity feature. Different from us, they only presented supervised approaches to rank the data based on these features.

3. Task Definition

3.1. Detection of previously fact-checked claims

One of the tasks that arise in the broader context of automated fact-checking is to check whether a claim has been fact-checked before. This can be considered the second step of a claim retrieval and verification pipeline, after the detection of check-worthy claims in different kinds of textual utterances and before the verification of those claims. This is addressed by task 2 [3]. More precisely, the task is to rank the most relevant verified claims out of a collection of already verified claims for a given input claim.

[1] https://github.com/Alihoe/CLEFCheckThat2aSimBa, https://github.com/Alihoe/CLEFCheckThat2bSimBa

3.2. Data

The subtasks cover two different types of media that are used to disseminate claims. Subtask A deals with tweets, subtask B with political debates and speeches. Both types of text sequences containing claim utterances will simply be referred to as *input claims* in the following. For both tasks different kinds of already fact-checked claims are made available. These will be called *verified claims*.[1]

Both input claims and verified claims consist of one or a few coherent sentences.

The input claims of subtask A are given as strings, divided into a training dataset of 1167 input claims, a development test dataset of 201 input claims and a final test dataset of 209 input claims. A human-annotated mapping from every input claim to the most relevant verified claim (query relevance or *qrels*-file) constitutes the gold standard. Verified claims are crawled from the fact-checking website *Snopes* and are provided in JSON format containing *title*, *subtitle*, *author*, *date* and a *vclaim*-entry with the content of the claim.

The input claims of subtask B are also provided as strings, divided into a training dataset of 702 input claims, a development test dataset of 79 input claims and a final test dataset of 65 input claims. Here, a human-annotated mapping from every input claim to one or more relevant verified claims is given in addition to the training data and as a gold standard for the test data. Furthermore, transcripts of the debates or speeches the input claims are obtained from are given for the test data. 19250 verified claims are taken from the fact-checking website *PolitiFact* and made available in JSON format containing the entries *vclaim_id*, *vclaim*, *date*, *truth_label*, *speaker*, *url*, *title* and *text*.

The mappings of input claims to verified claims will be referred to as *input-ver-claim pairs*.

4. Similarity-Based Features

4.1. Semantic Similarity

The task is formulated as a ranking-problem, where input-ver-claim-pairs are ranked depending on the relevance of the verified claim for fact-checking the input claim. Thus, the task can be considered a semantic textual similarity problem (STS) where sentences are compared by their semantic content to rank sentences containing similar claims highest (cf. [17]).

4.1.1. Sentence Embeddings

One promising way to deal with STS-problems is the usage of sentence embeddings. Sentence embeddings are fixed-sized vector representations that capture the meaning of sentences in so far that embeddings of semantically similar sentences are close in the corresponding vector space (cf. [18]). Sentence embedding models are usually trained on a huge amount of natural language data or rely on models that are trained on such. Thus they reflect the empirical distribution of linguistic elements and can be viewed as an appropriate method to investigate semantic similarity. That's because relying on the distributional hypothesis, "there is a correlation between distributional similarity and meaning similarity"[19].

The usefulness of the application of sentence embeddings has already been demonstrated by the participants of last year's lab. The sentence embedding model *Sentence-BERT*[9] was used

by the top-ranked teams of both subtask A and subtask B [16] [15]. Therefore, we use them as starting points for different components of our application.

Sentence-BERT (**SBERT**) is a modification of the transformer-based pre-trained language models *BERT*[20] or *RoBERTa*[8] using a Siamese network structure. The language models are trained on natural language inference (NLI) data and a pooling operation is added to their outputs in order to derive fixed-sized vector representations of the input sentences.

The idea of training on NLI data in a supervised way in order to get meaningful sentence embeddings was introduced by the authors of the sentence embedding model *InferSent*[18] (**InferSent**). However they did not build their model upon a tranformer-based language model, but on an encoder based on a bi-directional LSTM architecture fed with pre-trained word embeddings (*GloVe*[21] or *fastText*[22]).

Similarly, the model *Universal Sentence Encoder*[12] (**UniversalSE**) averages together word and bi-gram level embeddings, passes the representations through a feed-forward deep neural network (DNN) and is trained on NLI data.

The authors of *SimCSE*[23](**SimCSE**) also train their model on NLI data, but within a contrastive learning framework. Otherwise their model is similar to *Sentence-BERT*, relying on the pre-trained language models *BERT* and *RoBERTa* and adding a pooling operation to one of their output layers.

All sentence embedding models are also able encode small paragraphs instead of just sentences.

4.1.2. Measuring Semantic Similarity Using Sentence Embeddings

For all of these sentence embeddings methods, there are pre-trained models available that can be used out of the box. For *Sentence-BERT* we used *sentence-transformers/all-mpnet-base-v2*, because it performs best for STS tasks compared to the other pretrained models[2]. For InferSent we experimented with both versions, but report here only on the results obtained using version 2, which works with *fastText*[22], because we got better results than using the GloVe-vocabulary in pre-liminary experiments. For *Universal Sentence Encoder* we used *TF2.0 Saved Model (v4)*[3], because this is the most widely used model available for Universal Sentence Encoder and for *SimCSE* we used *princeton-nlp/sup-simcse-roberta-large*[4], because this also performs best for STS tasks compared to the other pretrained models [5].

Since sentence embeddings are vector representations of sentences within the same vector space, their similarity can be measured applying cosine similarity (**CosSim**), resulting in similarity scores which are rational numbers $\in$ [-100, 100]. These similarity scores should be referred to as **SentEmb**.

4.2. Other Measures of Similarity

In the following other measures of similarity are presented. An overview of their corresponding metrics can be found in Table 1.

[2]https://www.sbert.net/docs/pretrained_models.html
[3]https://tfhub.dev/google/universal-sentence-encoder/4
[4]https://huggingface.co/princeton-nlp/sup-simcse-roberta-large
[5]https://github.com/princeton-nlp/SimCSE

Table 1
Metrics of similarity features.

Kind of Similarity	Group	Feature	Metric
Semantic Similarity	SentEmb	SBERT	$\in$ [-100, 100]
		InferSent	
		UniversalSE	
		SimCSE	
String Similarity	LevDist	LevDist	$\in -\mathbb{Z}$
	StringSim	SeqMat	$\in [0, 1]$
		JaccChar	
		JaccTok	
Lexical Similarity	SimCount	WordCount	$\in \mathbb{N}$
	SimRatio	WordRatio	$\in [0, 100]$
		WordTokRatio	
Referential Similarity	SimCount	SynCount	$\in \mathbb{N}$
	SimRatio	SynRatio	$\in [0, 100]$
		SynTokRatio	
	SimCount	NE	$\in \mathbb{N}$
	SimRatio	NERatio	$\in [0, 100]$
		NeTokRatio	

4.2.1. String Similarity

In addition to the study of semantic similarity using sentence embeddings, there are other ways in which the similarity of sentences can be measured.

The most naive approach to measure the similarity of two sentences is to compare them at the string level, i.e. to see how far the characters and strings that make up a sentence differ from those of other sentences. We used three different methods to measure the string similarity of sentences: *Levenshtein Distance*, *Jaccard Distance* and *Sequence Matching*.

Levenshtein Distance (**LevDist**) is a metric to measure the distance between two strings by counting the number of operations (insertions, deletions or substitutions) needed to change one string into the other. Sentences which are similar thus have a small *Levenshtein Distance*. In order to adjust this distance score to the other similarity scores, such that a higher value signifies a higher similarity, we multiplied the Levenshtein Distance by *-1*. In practice, we thereby get negative three- or two-digit integers as similarity scores for almost all input-ver-claim pairs.

In general, *Jaccard Distance* is used to measure the similarity of sets. It is computed by dividing the size of the intersection by the size of the union of the sets. The closer this value is to one, the more similar are the sets. In context of sentence-similarity it can be applied in two ways: either regarding the characters (**JaccChar**) or the tokens (**JaccTok**) a sentence consists of as elements of a set.

The *Sequence Matching*-algorithm (**SeqMat**) provided by the Python library *difflib* works by comparing "the longest contiguous matching subsequence that contains no 'junk' elements" and recursively repeating this on the remaining subsequences. *Junk* elements are determined

heuristically based on the frequency of their duplicates in the text sequence [6].

Both the application of Jaccard Distance and Sequence Matching generate rational numbers $\in [0, 1]$. These similarity scores will be referred to as **StringSim**.

4.2.2. Lexical Similarity

Another type of similarity, which is not clearly distinguishable from semantic and string similarity, is lexical similarity or similarity of words. We used one method to capture lexical similarity between sentences and simply counted how often two claims contained the same words.

For this, we tokenized all claims using *NLTK's* word tokenizer[24], filtered out stop words and counted how often two claims contained the same tokens (**WordCount**). In order to value the number of equal tokens of shorter sentences higher than those of longer ones, we also computed a normalized ratio. For this we divided 100 by the number of tokens of both claims and multiplied the obtained value by two times the number of equal tokens.[7] We did this both including stop words (**WordTokRatio**) and not including them (**WordRatio**).

Counting equal tokens we gained a positive integer similarity score, usually with less than three digits. We call this kind of discrete score **SimCount**. Computing the ratios we obtained percentages similar to the **SentEmb**-scores $\in [0, 100]$. This kind of scores will be referred to as **SimRatio**.

4.2.3. Referential Similarity

Another way to think of similarity between sentences is to examine whether they refer to the same objects. To represent this kind of similarity we used two methods. Similar to the lexical similarity approach, we counted how often two claims contained words which are synonyms of each other. Additionally, we counted how often two claims contain the same named entities (*NEs*).

To compare the synonyms, we used WordNet[25] and looked for all available synsets the tokens mentioned in a claim are part of. We tokenized the sentences the same way as above. Then we counted how often two claims contained the same synsets (**SynCount**). Here we also computed the ratio of the count of synonyms regarding all synonyms (**SynRatio**) and all tokens (**SynTokRatio**) in the two sentences.

In order to compare *NEs* we used the entity-fishing system[26], which recognizes named entities mentioned in a text and disambiguates them using Wikidata. The system is able to return the the Wikipedia and Wikidata identifiers of those mentions. We counted how often two claims contained named entities related to the same Wikipedia or Wikidata entry (**NE**). We also additionally computed the ratio of the count of *NEs* regarding all *NEs* (**NERatio**) and all tokens (**NETokRatio**) in the two sentences.

Similarly to the lexical similarity scores, we obtained two different kinds of metrics for these similarities: **SimCount** and **SimRatio** (see Table 1).

[6] https://docs.python.org/3/library/difflib.html

[7] e.g.: If two claims consisted of ten tokens each and had ten tokens in common, we would obtain a **WordTokRatio** of *(100/20)*10*2 =* ***100***. If they only had one token in common the obtained ratio would be *(100/20)*1*2 =* ***10***. If both claims consisted of 50 tokens each, the obtained ratios would be *(100/100)*10*2 =* ***20*** and *(100/20)*1*2 =* ***2***.

4.3. Pre-Processsing

4.3.1. Cleaning tweets

For both subtasks we experimented with different ways of pre-processing the input claims. We cleaned the tweets given in subtask 2a to get rid of redundant information. We removed URLs, @-symbols and user-information (see Table 2).

Table 2
Example of a cleaned tweet from the test data.

Original Tweet	Cleaned Tweet
Starlink — here. Thanks, @elonmusk pic.twitter.com/dZbaYqWYCf — Mykhailo Fedorov (@FedorovMykhailo) February 28, 2022	*Starlink — here. Thanks, elonmusk*

4.3.2. Including context

For subtask 2b, we tried incorporating the input claims' contexts within the speech or debate they were obtained from. We included the lines that were spoken before and after the relevant claim and integrated information about the current speaker by prepending "speaker X said" to the line of speech, where X is substituted by the name of the respective speaker (see Table 3).

Table 3
Example of a contextualized claim from the development test data.

Original Input Claim	Contextualized Input Claim
He wouldn't send anything else.	*donald trump said "And Obama would send pillows and sheets." donald trump said "He wouldn't send anything else." donald trump said "It's the whole thing."*

5. Model

5.1. Unsupervised Approach

We tried out an unsupervised and a supervised method to utilize the information we gained on the different kinds of similarity. The main idea of the unsupervised approach is to rank the input-ver-claim pairs by the different similarity scores described above. Therefore a general similarity score is computed, combining the varying metrics (see Table 1). This general score can roughly be compared to the percentage to which two sentences are similar where two exactly equal sentences would have a score of roughly 100. However, our way of combining the different similarity scores does not ensure that the resulting score is smaller than 100. It can sometimes be a little higher.

The general similarity score is computed the following way:

- taking the mean of all *SentEmb*-, *SimRatio*- and *StringSim*-scores normalized to [0, 100]
- incorporating the *LevDist*: First the *LevDist* is divided by -100, which generates a positive factor that is smaller the more similar two sentences are. Then the similarity score obtained by computing the mean, is divided by this factor. [8]
- adding the *SimCount*-scores to the obtained score

For the output the five most similar verified claims for an input claim are computed relying on the general similarity score.

5.2. Supervised Approach

For the supervised approach we built a feature set out of the different similarity scores in order to classify if a verified claim is relevant for an input claim. We experimented with different methods to optimize our classification results. We used *Blocking* and *Balancing* in order to optimize our training results. Additionally we tried out different *Classifiers* and applied *Feature Selection* to further improve our output. Lastly we also made use of a *heuristic* based on our supervised approach to find relevant verified claims for all input claims.

To optimize the training, we used a *Blocking* approach. Instead of generating negative training instances by pairing each input claim with all but the true matching verified claims in the dataset, we computed the 50 most similar verified claims according to either of the four *SentEmb* scores and generated negative training instances using only those. More specifically, we extracted 4 sets of input-ver-claim-pairs, one set for each *SentEmb* method, with each set containing the 50 most similar verified claims identified by this method. Then we used the union of these sets as our final training set. We observed that all true input-ver-claims were covered. Besides the computational advantage of a smaller training set, this way the model may better learn to distinguish cases that are similar on the surface as all very dissimilar pairs have been filtered out before training.

Then all similarity scores, (also the *SentEmb*-scores) were added as features. As targets we obtained the relevance scores from the *qrels*-file of the training data. An unlabeled feature set was built for the test data.

After blocking, the percentage of true positives in our training data was still beneath 1% for both subtasks. That's why we applied *Random Undersampling* as a *Balancing* method and experimented with different parameters (see Tables 4 and 5).

Then a *Classifier* was trained on the training data to predict relevance scores for the test data. We also experimented with different classifiers suited for binary classification, such as *KNN*, *Logistic Regression*, *Linear SVC* and a *Decision Tree* (see Tables 6, 7).

We experimented with different *selections of features* out of the similarity features presented above. The influence of the ensemble of features is shown in Tables 13 and 14. Additionally we

[8] e.g.: Given is a SentEmb mean of *50.0*. If two sentences consist of quite similar strings, one could imagine them having a *LevDist* of *-50*. If two sentences are not that similar, they could have a *LevDist* of *-200*. Applying the technique described, incorporating *LevDist* would result in the sim score *100* for the similar sentences and *25* for the varying sentences. This way it is not ensured that the obtained similarity score is $\in [0, 100]$. In practice, however, the calculated values are in this range.

Table 4
Subtask 2A: Impact of balancing using KNN.

Positives	Selected Features	Classified Positives	MAP@5
0.62 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **Lexical Similarity:** WordCount, WordTokRatio	138	**0.8865**
1 %	"	196	**0.8865**
2 %	"	247	0.8760
3 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **Lexical Similarity:** WordCount, WordRatio, WordTokRatio	287	0.8664
4 %	"	340	0.8712
5 %	"	373	0.8784
6 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **Lexical Similarity:** WordRatio, WordTokRatio	402	0.8656
7 %	"	433	0.8652
8 %	"	468	0.8628
9 %	"	486	0.8628
10 %	"	520	0.8700
20 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **Lexical Similarity:** WordRatio	714	0.8776
30 %	**Semantic Similarity:** SBERT, InferSent, SimCSE	973	0.8836
40 %	"	1158	0.8884
50 %	"	1368	**0.8896**
60 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **String Similarity:** LevDist	1532	0.8805
70 %	"	1660	0.8829
80 %	"	1852	0.8805
90 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **String Similarity:** LevDist **Referential Similarity:** SynCount, SynTokRatio	2111	**0.8896**
100 %	**Semantic Similarity:** SBERT, InferSent, SimCSE **String Similarity:** LevDist **Referantial Similarity:** SynCount	2102	0.8713

included the feature *TokenCount* which represents the sum of tokens of both input claim and verified claim.

If no relevant verified claim was predicted for an input claim, we relied on our unsupervised approach *heuristically* and chose the five most similar verified claims based on the mean of sentence embedding similarity scores. For 2A we chose *SBERT, InferSent and SimCSE* as *SentEmb* scores, for 2B all four models, including *UniversalSE*.

6. Results

6.1. Evaluation Metric

The task is considered a ranking task and is evaluated as such. The official ranking evaluation measure is *Mean Average Precision at 5* (**MAP@5**). Additionally the provided scorer computes the measures MAP@k for k=1, 3, 5, 10, MRR and Precision@k for k= 3, 5, 10 (cf. [3]). The MAP@k metric measures the mean of correctly classified pairs in the top k of the returned output. MRR or *Mean Reciprocal Rank* measures how far the assigned rank of a correct pair

Table 5
Subtask 2B: Impact of balancing using Logistic Regression.

Positives	Selected Features	Classifed Positives	MAP@5
0.65 %	**Semantic Similarity:** SimCSE **String Similarity:** LevDist **Lexical Similarity:** WordCount, WordRatio, WordTokRatio TokenCount	11	**0.4721**
1 %	"	15	0.4669
2 %	**Semantic Similarity:** SimCSE **String Similarity:** LevDist **Lexical Similarity:** WordCount, WordRatio, WordTokRatio **Referential Similarity:** SynTokRatio TokenCount	22	0.4669
3 %	**Semantic Similarity:** SimCSE **String Similarity:** LevDist **Lexical Similarity:** WordCount, WordRatio, WordTokRatio TokenCount	26	0.4503
4 %	"	34	0.4579
5 %	"	43	0.4464
6 %	**Semantic Similarity:** SimCSE **String Similarity:** LevDist **Lexical Similarity:** WordRatio, WordTokRatio TokenCount	61	0.4531
7 %	"	68	0.4531
8 %	"	82	**0.4608**
9 %	"	92	0.4608
10 %	"	106	0.4454
20 %	**Semantic Similarity:** SimCSE **String Similarity:** LevDist **Lexical Similarity:** WordRatio, WordTokRatio **Referential Similarity:** SynRatio TokenCount	258	0.4569
30 %	**Semantic Similarity:** SimCSE **Lexcial Similarity:** WordRatio, WordTokRatio **Referential Similarity:** SynTokRatio	453	0.4436
40 %	"	637	0.4359
50 %	"	809	**0.4332**
60 %	"	981	0.4324
70 %	"	1171	0.4436
80 %	**Semantic Similarity:** SimCSE **Lexical Similarity:** WordCount, WordTokRatio **Referential Similarity:** SynCount, SynTokRatio	1265	0.4436
90 %	"	1393	0.4551
100 %	**Semantic Similarity:** SimCSE **Lexcial Similarity:** WordRatio, WordTokRatio **Referential Similarity:** SynCount, SynTokRatio	1547	0.4551

differs from its correct rank (i.e. the first rank for subtask A) on average.

Table 6
Subtask 2A: Impact of Classifier without balancing using features *SBERT, InferSent, SimCSE, WordCount, WordTokRatio* and with balancing to 50% positives using features *SBERT, InferSent, SimCSE*.

Classifier	MAP@5 No Balancing	MAP@5 with Balancing
KNN	**0.8865**	**0.8896**
Logistic Regression	0.8832	0.8844
Linear SVC	0.8792	0.8820
Decision Tree	0.8502	0.8478

Table 7
Subtask 2B: Impact of classifier with balancing to 50% Positives using features SimCSE, SynTokRatio, WordRatio, WordTokRatio and to 8% Positives using features *SimCSE, LevDist, WordRatio, WordTokRatio, TokenCount*.

Classifier	MAP@5 with 50% Positives	MAP@5 with 8% Positives
KNN	0.4328	0.4179
Logistic Regression	0.4332	**0.4608**
Linear SVC	**0.4340**	0.4485
Decision Tree	0.4136	0.3538

6.2. Subtask 2A

For Subtask 2A we got the best result with our unsupervised approach, combining the similarity scores of *SBERT, SimCSE, WordCount* and *WordTokRatio* with a MAP@5 of **0.9175** (see Table 13).

However the output we submitted made use of *SBERT, SimCSE* and *WordCount* and scored slightly worse (**0.9075**) (see Table 8). We still achieved a score above the baselines utilizing a simple and fast unsupervised ranking method.

Table 8
Subtask 2A: Results.

User/Team	MAP@5	P@5	RR
mshlis	0.956	0.322	0.957
watheq9	0.921	0.189	0.923
Viktor	0.922	0.190	0.922
Team_SimBa	**0.907**	0.190	0.907
motlogelwan	0.873	0.187	0.878
fraunhofersit_checkthat22	0.610	0.141	0.624
Team_Vax_Misinfo	0.020	0.011	0.096
Random Baseline	0	0	0
BM25 Baseline	0.8179		

6.3. Subtask 2B

For Subtask 2B we got the best results using a supervised approach. All similarity features were included, except from *JaccChar* (see Table 14). We made use of **Random Undersampling** to increase the percentage of positives in the training data (relevant input-ver-claim pairs) to 8%. Then a **Logistic Regression Classifier** was trained and predicted 111 input-ver-claim pairs. The unsupervised heuristic described above was used to find relevant verified claims for the remaining input claims. This way the output achieved a MAP@5 of **0.4882**.

The output we submitted also scored slightly worse than our best result with a MAP@5 of **0.459**. To generate this output we used **Linear Support Vector Classification** and sampled to 14% positives. The considered features were *SimCSE, JaccTok, WordCount, WordRatio, SynCount* and *SynRatio*. This is still the best result for subtask 2B (see Table 9).

Table 9
Subtask 2B: Results.

User/Team	MAP@5	P@5	RR
Team_SimBa	**0.459**	0.126	0.475
Team_Vax_Misinfo	0.091	0.040	0.131
Random Baseline	0	0	0
BM25 Baseline	0.3207		

6.4. Result of Pre-Processing

It turned out that our pre-processing approach did not improve our results on the test data for Subtask 2A (see Table 10), although it did for the development test data. This is an issue worth investigating in future work. Tweet-specific units of text such as user-information were removed and it showed that it would have been useful to incorporate this kind of information for solving the task 2A. Nevertheless the pre-processing ensured that the data of both tasks was more similar and thereby helped assessing similarity of claims in general contexts.

The incorporation of context for subtask 2B also did not improve the results on the development test data and on the final test data. That is why we used the original data for subtask 2B.

Table 10
Subtask 2A: Impact of pre-processing.

	MAP@5
with Pre-Processing	0.9143
without Pre-Processing	0.9270

7. Observations

7.1. Evaluation of Features

7.1.1. Powerful Features for Subtask A and Subtask B

Table 11
Supervised Approach: Comparison of performance of similarity scores independently.

Similarity Scores	MAP@5 Subtask 2A	MAP@5 Subtask 2B
CosSim SBERT	**0.8711**	0.3664
CosSim InferSent	0.4208	0.1846
CosSim UniversalSE	0.7153	0.3872
CosSim SimCSE	0.7973	**0.3946**
LevDist	0.1271	**0.1833**
JaccChar	0.0522	0.0569
JaccTok	0.4014	0.2763
SeqMat	0.2698	**0.2790**
WordCount	0.5667	0.2731
WordRatio	0.6454	0.2967
WordTokRatio	0.6630	0.2954
SynCount	0.3228	0.2024
SynRatio	0.3196	0.2508
SynTokRatio	0.3071	0.2359
NE	0.4549	0.1600
NERatio	0.4357	0.1556
NETokRatio	0.4620	0.1654

The observation of the results of using the supervised approach on single features (see Table 11) gives a good overview of their independent performance. As expected, the most successful features for both subtasks are the cosine similarities of the sentence embeddings. Especially *SBERT, UniversalSE* and *SimCSE* performed best on both task. That's because, as explained above, sentence embeddings are really useful to capture STS.

Interestingly *SBERT* is the most powerful feature for Subtask 2A and *SimCSE* the most powerful one for Subtask 2B. It would be worth further investigations to identify the reason for this difference. Both models are pre-trained on a large share of the same data, so maybe the contrastive training objective of *SimCSE* is partly responsible for it.

Another important observation is the fact that the lexical similarity features *WordCount, WordRatio* and *WordTokRatio* perform also really well for both tasks. This is kind of surprising, because these features are generated in such a simple way.

In contrast, the Jaccard Similarity of characters *JaccChar* is the weakest similarity feature. This can be explained by the fact that the consideration of equal characters, regardless of their order, doesn't have much informational value for the meaning of a sentence as a whole.

One interesting finding regarding the differences between the subtasks is the varying performance of string similarity features. The string similarity features *LevDist* and *SeqMat* are the only features that produce a higher MAP@5 for Subtask 2B than for Subtask 2A. Looking at the

data, it is noticeable that the input claims and the verified claims provided for Subtask 2B often share long, continuous strings (see Table 12).

Table 12
Comparison of input-ver-claim-pairs of subtask A and subtask B. The claims of subtask B share a long, continuous string.

	Input Claim	Verified Claim
Subtask 2A	*TIME's new cover: How Putin shattered Europe's dreams*	*Time magazine compared Russian President Vladimir Putin to Adolf Hitler on the cover of the March 14 / March 21, 2022, issue.*
Substask 2B	***160 million people like their private insurance**, and if they don't like it, they can buy into a Medicare-like proposal..*	***160 million people like their private insurance**.*

7.1.2. Feature Set

One of the most intriguing observations is the fact that both the unsupervised and the supervised approach perform best if lexical similarity is considered besides semantic similarity (see Tables 13 and 14). The *SentEmb* features do not seem to cover lexical similarity and their performance benefits from the additional information contained by lexical similarity features. This is also supported by the observation that these two types of features do not have a strong correlation (see Tables 15 and 16).

Also it can be observed that especially for subtask B it is helpful to consider the combination of almost all similarity features in the supervised approach (see Table 14).

Overall a higher number of features mostly increases the performance of the supervised approach and decreases it for the unsupervised approach as relatively uninformative features have a too high impact on the latter.

7.2. Supervised vs Unsupervised Approach

One important observation with respect to our results is the fact that the unsupervised approach performs nearly as good as the unsupervised approach for subtask B and even better than the unsupervised approach for subtask A.

Since the task is a ranking problem, the unsupervised approach seems to perform sufficiently well for the given task. For similar tasks with the constraint to only find pairs that are relevant with a high certainty, the supervised approach might be more helpful.

Also it is reasonable to assume that the unsupervised approach generalizes well over similar tasks, because it is independent of the training data. This assumption is supported by the fact that the features that produce the best outputs are almost the same for both subtask A and

Table 13
Subtask 2A: Comparison of the combination of different similarity scores in a supervised and unsupervised way.

Kinds of Similarity Scores	Similarity Scores	MAP@5 UnSup	MAP@5 KNN 50% positives	MAP@5 KNN no balancing
Semantic Similarity	SBERT, InferSent, UniversalSE, SimCSE	0.8883	0.8734	0.8672
	SBERT, UniversalSE, SimCSE	0.8972	0.8748	0.8829
	SBERT, InferSent, SimCSE	0.8793	0.8896	0.8664
	SBERT, SimCSE	0.8896	0.8781	0.8748
Semantic Similarity and Lexical Similarity	SBERT, SimCSE, WordCount	*0.9075*	0.8839	0.8792
	SBERT, SimCSE, WordTokRatio	0.9151	0.8877	0.8792
	SBERT, SimCSE, WordCount, WordTokRatio	**0.9175**	**0.8955**	0.8801
	SBERT, InferSent, SimCSE, WordCount, WordTokRatio	0.8911	0.8832	**0.8865**
	SBERT, InferSent, SimCSE, WordCount	0.8941	0.8817	0.8780
Semantic Similarity, Lexical Similarity and Referential Similarity	SBERT, SIMCSE, WordTokRatio, NETokRatio	0.9172	0.8863	0.8825
Semantic Similarity, String Similarity and Lexical Similarity	SimCSE, LevDist, all WordSims	0.8027	0.8521	0.8742
	SimCSE, LevDist, WordRatio, WordTokRatio	0.7986	0.8305	0.8670
Semantic Similarity, String Similarity, Lexical Similarity and Referential Similarity	SBERT, WordCount, JaccTok, NETokRatio	0.8929	0.8748	0.8744
	SBERT, WordTokRatio, JaccTok, NETokRatio	0.9001	0.8720	0.8844
	SimCSE, JaccTok, all WordSims, all SynSims	0.5509	0.8473	0.8650
	SimCSE, SeqMat, JaccTok, all WordSims, all SynSims	0.5490	0.8417	0.8518
	SimCSE, SeqMat, JaccTok, all WordSims, SynRatio, SynTokRatio	0.6540	0.8323	0.8602
	SimCSE, LevDist, all WordSims, SynTokRatio	0.7448	0.8628	0.8550
	SimCSE, LevDist, WordRatio, WordTokRatio, SynRatio	0.7425	0.8501	0.8778
	SimCSE, SynTokRatio, WordRatio, WordTokRatio	0.7030	0.8573	0.8610
	SimCSE, WordCount, WordTokRatio, SynCount, SynTokRatio	0.5850	0.8537	0.8650
	SimCSE, SynCount, SynTokRatio, WordRatio, WordTokRatio	0.5842	0.8585	0.8586
ALL	ALL Except JaccChar, NERatio	0.6323	0.8521	0.8754
	ALL Except JaccChar	0.6540	0.8620	0.8754
	ALL	0.6376	0.8642	0.8793

Table 14
Subtask 2B: Comparison of the combination of different similarity scores in a supervised and unsupervised way.

Kinds of Similarity Scores	Similarity Scores	MAP@5 UnSup	MAP@5 LogReg 8% positives	MAP@5 LinearSVC
Semantic Similarity	SBERT, InferSent, UniversalSE, SimCSE	**0.4721**	0.4190	0.4454
	SBERT, UniversalSE, SimCSE	0.4672	0.4190	0.4454
	SBERT, InferSent, SimCSE	0.4310	0.4190	0.4454
	SBERT, SimCSE	0.4190	0.4344	0.4454
Semantic Similarity and Lexical Similarity	SBERT, SimCSE, WordCount	0.4395	0.4554	0.4531
	SBERT, SimCSE, WordTokRatio	0.4654	0.4479	0.4537
	SBERT, SimCSE, WordCount, WordTokRatio	**0.4718**	0.4479	0.4332
	SBERT, InferSent, SimCSE, WordCount, WordTokRatio	0.4654	0.4479	0.4332
	SBERT, InferSent, SimCSE, WordCount	0.4583	0.4554	0.4562
Semantic Similarity, Lexical Similarity and Referential Similarity	SBERT, SIMCSE, WordTokRatio, NETokRatio	0.4190	0.4479	**0.4691**
Semantic Similarity, String Similarity, Lexical Similarity and Referential Similarity	SBERT, WordCount, JaccTok, NETokRatio	0.4595	0.4056	0.4590
	SBERT, WordTokRatio, JaccTok, NETokRatio	0.4415	0.4338	0.4295
	SimCSE, JaccTok, all WordSims, all SynSims	0.3205	0.4646	0.4308
	SimCSE, SeqMat, JaccTok, all WordSims, all SynSims	0.3195	0.4646	0.4308
	SimCSE, SeqMat, JaccTok, all WordSims, SynRatio, SynTokRatio	0.3367	0.4646	0.4308
Semantic Similarity, String Similarity and Lexical Similarity	SimCSE, LevDist, all WordSims	0.3731	0.4608	0.4428
	SimCSE, LevDist, all WordSims, SynTokRatio	0.3477	0.4646	0.4308
	SimCSE, LevDist, WordRatio, WordTokRatio	0.3641	0.4608	0.4569
	SimCSE, LevDist, WordRatio, WordTokRatio, SynRatio	0.3542	0.4646	0.4340
	SimCSE, SynTokRatio, WordRatio, WordTokRatio	0.3355	0.4646	0.4340
	SimCSE, WordCount, WordTokRatio, SynCount, SynTokRatio	0.3118	0.4531	0.4269
	SimCSE, SynCount, SynTokRatio, WordRatio, WordTokRatio	0.3301	0.4646	0.4385
ALL	ALL Except JaccChar, NERatio	0.3301	0.4869	0.4436
	ALL Except JaccChar	0.3147	**0.4882**	0.4436
	ALL	0.3147	0.4749	0.4513

subtask B for the unsupervised approach (see Tables 13 and 14), while the supervised approach relies on different features for the subtasks to produce good outputs.

8. Future Work

It would be interesting to investigate the generalizability of our approach and to check if the assumption that the unsupervised approach generalizes better than the supervised approach is true. Also a detailed assessment of the impact of pre-processing would be beneficial for related works.

9. Conclusion

We treated the task to detect previously fact-checked claims as a STS-task. To solve it, we investigated different kinds of similarity measures between sentences, covering semantic, lexical and referential similarity. We found that it is beneficial to combine semantic similarity measures gained by calculating the distance of sentence embeddings with lexical similarity measures gained by counting shared words. Furthermore, we found that an unsupervised approach can be even more successful than a supervised approach for this task. Overall, our proposed approaches provide very good results for both subtasks with a MAP@5 of 0.907 for subtask A and a MAP@5 of 0.459 for subtask B, both scoring above the baselines and even being the top-ranked output for subtask B.

References

[1] P. Nakov, G. Da San Martino, F. Alam, S. Shaar, H. Mubarak, N. Babulkov, Overview of the CLEF-2022 CheckThat! lab task 2 on detecting previously fact-checked claims, in: Working Notes of CLEF 2022—Conference and Labs of the Evaluation Forum, CLEF '2022, Bologna, Italy, 2022.

[2] S. Shaar, A. Nikolov, N. Babulkov, F. Alam, A. Barrón-Cedeño, T. Elsayed, M. Hasanain, R. Suwaileh, F. Haouari, G. D. S. Martino, P. Nakov, Overview of checkthat! 2020 english: Automatic identification and verification of claims in social media., in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), CLEF (Working Notes), volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_265.pdf.

[3] S. Shaar, F. Haouari, W. Mansour, M. Hasanain, N. Babulkov, F. Alam, G. Da San Martino, T. Elsayed, P. Nakov, Overview of the CLEF-2021 CheckThat! lab task 2 on detecting previously fact-checked claims in tweets and political debates, in: Working Notes of CLEF 2021—Conference and Labs of the Evaluation Forum, CLEF '2021, Bucharest, Romania (online), 2021. URL: http://ceur-ws.org/Vol-2936/paper-29.pdf.

[4] M. Bouziane, H. Perrin, A. Cluzeau, J. Mardas, A. Sadeq, Team buster.ai at checkthat! 2020 insights and recommendations to improve fact-checking, in: L. Cappellato, C. Eickhoff, N. F. 0001, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation

Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_134.pdf.

[5] E. Thuma, N. P. Motlogelwa, T. Leburu-Dingalo, M. Mudongo, Ub_et at checkthat! 2020: Exploring ad hoc retrieval approaches in verified claims retrieval, in: L. Cappellato, C. Eickhoff, N. F. 0001, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_204.pdf.

[6] T. McDonald, Z. Dong, Y. Zhang, R. Hampson, J. Young, Q. Cao, J. L. Leidner, M. Stevenson, The university of sheffield at checkthat! 2020: Claim identification and verification on twitter, in: L. Cappellato, C. Eickhoff, N. F. 0001, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_162.pdf.

[7] G. S. Cheema, S. Hakimov, R. Ewerth, Check square at checkthat! 2020: Claim detection in social media via fusion of transformer and syntactic features, in: L. Cappellato, C. Eickhoff, N. F. 0001, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_216.pdf.

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. URL: https://arxiv.org/abs/1907.11692. doi:10.48550/ARXIV.1907.11692.

[9] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. arXiv:1908.10084.

[10] L. C. Passaro, A. Bondielli, A. Lenci, F. Marcelloni, Unipi-nle at checkthat! 2020: Approaching fact checking from a sentence similarity perspective through the lens of transformers, in: L. Cappellato, C. Eickhoff, N. F. 0001, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_169.pdf.

[11] U. Shukla, A. Sharma, Tiet at clef checkthat! 2020: Verified claim retrieval, in: L. Cappellato, C. Eickhoff, N. F. 0001, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_197.pdf.

[12] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, R. Kurzweil, Universal sentence encoder, CoRR abs/1803.11175 (2018). URL: http://arxiv.org/abs/1803.11175. arXiv:1803.11175.

[13] J. Martinez-Rico, L. Araujo, J. Martinez-Romo, Nlpir@uned at checkthat! 2020: A preliminary approach for check-worthiness and claim retrieval tasks using neural networks and graphs, 2020.

[14] A. Pritzkau, Nlytics at checkthat! 2021: Detecting previously fact-checked claims by measuring semantic similarity, in: Working Notes of CLEF 2021—Conference and Labs of

the Evaluation Forum, CLEF '2021, Bucharest, Romania (online), 2021. URL: http://ceur-ws.org/Vol-2936/paper-47.pdf.
[15] S. Mihaylova, I. Borisova, D. Chemishanov, P. Hadzhitsanev, M. Hardalov, P. Nakov, Dips at checkthat! 2021: Verified claim retrieval, in: Working Notes of CLEF 2021—Conference and Labs of the Evaluation Forum, CLEF '2021, Bucharest, Romania (online), 2021. URL: http://ceur-ws.org/Vol-2936/paper-45.pdf.
[16] A. Chernyavskiy, D. Ilvovsky, P. Nakov, Aschern at checkthat! 2021: Lambda-calculus of fact-checked claims, in: Working Notes of CLEF 2021—Conference and Labs of the Evaluation Forum, CLEF '2021, Bucharest, Romania (online), 2021. URL: http://ceur-ws.org/Vol-2936/paper-38.pdf.
[17] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, SemEval-2012 task 6: A pilot on semantic textual similarity, in: *SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), Association for Computational Linguistics, Montréal, Canada, 2012, pp. 385–393. URL: https://aclanthology.org/S12-1051.
[18] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, Supervised learning of universal sentence representations from natural language inference data, 2017. URL: https://arxiv.org/abs/1705.02364. doi:10.48550/ARXIV.1705.02364.
[19] M. Sahlgren, The distributional hypothesis, The Italian Journal of Linguistics 20 (2008) 33–54.
[20] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423. doi:10.18653/v1/N19-1423.
[21] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.
[22] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, arXiv preprint arXiv:1607.04606 (2016).
[23] T. Gao, X. Yao, D. Chen, SimCSE: Simple contrastive learning of sentence embeddings, in: Empirical Methods in Natural Language Processing (EMNLP), 2021.
[24] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.
[25] C. Fellbaum, WordNet: An Electronic Lexical Database, Bradford Books, 1998.
[26] P. Lopez, entity-fishing, https://github.com/kermitt2/entity-fishing, 2016–2022. arXiv:1:dir:cb0ba3379413db12b0018b7c3af8d0d2d864139c.

A. Appendix

Table 15
Subtask 2A: Spearman correlation between different similiarity scores for the test data.

SimScore	SBERT	InferSent	UniversalSE	SimCSE	LevDist	JaccChar	JaccTok	SeqMat
SBERT	1.0	-0.1	**0.43**	**0.51**	0.06	0.0	0.02	0.05
InferSent	-0.1	1.0	0.09	-0.09	-0.54	**0.38**	0.44	-0.36
UniversalSE	0.43	0.09	1.0	0.36	0.08	0.05	0.15	0.08
SimCSE	**0.51**	-0.09	0.36	1.0	0.05	0.01	0.02	0.03
LevDist	0.06	-0.54	0.08	0.05	1.0	-0.28	-0.23	**0.74**
JaccChar	0.0	0.38	0.05	0.01	-0.28	1.0	0.37	-0.14
JaccTok	0.02	0.44	0.15	0.02	-0.23	0.37	1.0	-0.06
SeqMat	0.05	-0.36	0.08	0.03	**0.74**	-0.14	-0.06	1.0
WordCount	0.1	**0.51**	0.19	0.07	-0.43	0.37	**0.67**	-0.28
WordRatio	0.17	0.24	0.24	0.13	-0.12	0.29	0.63	-0.04
WordTokRatio	0.19	0.2	0.26	0.15	-0.05	0.25	0.59	0.02
SynCount	0.07	0.5	0.16	0.05	-0.42	0.31	0.48	-0.27
SynRatio	0.14	0.28	0.23	0.12	-0.14	0.22	0.45	-0.05
SynTokRatio	0.14	0.26	0.23	0.12	-0.11	0.21	0.42	-0.04
NE	0.28	0.11	0.38	0.24	-0.0	0.09	0.23	0.02
NERatio	0.28	0.1	0.38	0.24	0.0	0.09	0.23	0.02
NeTokRatio	0.28	0.1	0.38	0.24	0.01	0.08	0.23	0.03

SimScore	WordCount	WordRatio	WordTokRatio	SynCount	SynRatio	SynTokRatio
SBERT	0.1	0.17	0.19	0.07	0.14	0.14
InferSent	0.51	0.24	0.2	0.5	0.28	0.26
UniversalSE	0.19	0.24	0.26	0.16	0.23	0.23
SimCSE	0.07	0.13	0.15	0.05	0.12	0.12
LevDist	-0.43	-0.12	-0.05	-0.42	-0.14	-0.11
JaccChar	0.37	0.29	0.25	0.31	0.22	0.21
JaccTok	0.67	0.63	0.59	0.48	0.45	0.42
SeqMat	-0.28	-0.04	0.02	-0.27	-0.05	-0.04
WordCount	1.0	0.9	0.86	**0.72**	0.64	0.62
WordRatio	0.9	1.0	0.98	0.6	**0.67**	**0.64**
WordTokRatio	0.86	0.98	1.0	0.57	0.66	**0.64**
SynCount	**0.72**	0.6	0.57	1.0	0.91	0.92
SynRatio	0.64	**0.67**	**0.66**	0.91	1.0	0.97
SynTokRatio	0.62	0.64	0.64	0.92	0.97	1.0
NE	0.32	0.36	0.36	0.2	0.24	0.23
NERatio	0.31	0.36	0.36	0.19	0.24	0.23
NeTokRatio	0.31	0.36	0.36	0.19	0.24	0.23

SimScore	NE	NERatio	NETokRatio
SBERT	0.28	0.28	0.28
InferSent	0.11	0.1	0.1
UniversalSE	**0.38**	**0.38**	**0.38**
SimCSE	0.24	0.24	0.24
LevDist	-0.0	0.0	0.01
JaccChar	0.09	0.09	0.08
JaccTok	0.23	0.23	0.23
SeqMat	0.02	0.02	0.03
WordCount	0.32	0.31	0.31
WordRatio	0.36	0.36	0.36
WordTokRatio	0.36	0.36	0.36
SynCount	0.2	0.19	0.19
SynRatio	0.24	0.24	0.24
SynTokRatio	0.23	0.23	0.23
NE	1.0	1.0	1.0
NERatio	1.0	1.0	1.0
NeTokRatio	1.0	1.0	1.0

Table 16
Subtask 2B: Spearman correlation between different similiarity scores for the test data.

SimScore	SBERT	InferSent	UniversalSE	SimCSE	LevDist	JaccChar	JaccTok	SeqMat
SBERT	1.0	0.26	0.59	0.61	-0.06	0.21	0.04	0.04
InferSent	0.26	1.0	0.39	0.15	-0.59	0.47	0.38	-0.14
UniversalSE	0.59	0.39	1.0	0.48	-0.09	0.28	0.25	0.09
SimCSE	0.61	0.15	0.48	1.0	-0.01	0.19	0.1	0.05
LevDist	-0.06	-0.59	-0.09	-0.01	1.0	-0.23	-0.03	0.57
JaccChar	0.21	0.47	0.28	0.19	-0.23	1.0	0.33	0.06
JaccTok	0.04	0.38	0.25	0.1	-0.03	0.33	1.0	0.21
SeqMat	0.04	-0.14	0.09	0.05	0.57	0.06	0.21	1.0
WordCount	0.25	0.6	0.43	0.2	-0.33	0.36	0.59	-0.01
WordRatio	0.11	0.11	0.28	0.14	0.23	0.16	0.59	0.28
WordTokRatio	0.16	0.04	0.29	0.16	0.33	0.08	0.5	0.33
SynCount	0.31	0.57	0.46	0.22	-0.37	0.32	0.41	-0.03
SynRatio	0.26	0.3	0.4	0.21	-0.01	0.18	0.43	0.16
SynTokRatio	0.27	0.29	0.42	0.22	0.0	0.18	0.4	0.18
NE	0.34	0.17	0.43	0.3	-0.07	0.18	0.21	0.05
NERatio	0.34	0.16	0.43	0.3	-0.05	0.17	0.21	0.06
NeTokRatio	0.34	0.16	0.43	0.3	-0.05	0.17	0.21	0.07

SimScore	WordCount	WordRatio	WordTokRatio	SynCount	SynRatio	SynTokRatio
SBERT	0.25	0.11	0.16	0.31	0.26	0.27
InferSent	0.6	0.11	0.04	0.57	0.3	0.29
UniversalSE	0.43	0.28	0.29	0.46	0.4	0.42
SimCSE	0.2	0.14	0.16	0.22	0.21	0.22
LevDist	-0.33	0.23	0.33	-0.37	-0.01	0.0
JaccChar	0.36	0.16	0.08	0.32	0.18	0.18
JaccTok	0.59	0.59	0.5	0.41	0.43	0.4
SeqMat	-0.01	0.28	0.33	-0.03	0.16	0.18
WordCount	1.0	0.76	0.7	0.74	0.64	0.63
WordRatio	0.76	1.0	0.96	0.45	0.63	0.61
WordTokRatio	0.7	0.96	1.0	0.4	0.6	0.62
SynCount	0.74	0.45	0.4	1.0	0.87	0.9
SynRatio	0.64	0.63	0.6	0.87	1.0	0.95
SynTokRatio	0.63	0.61	0.62	0.9	0.95	1.0
NE	0.37	0.32	0.32	0.27	0.26	0.25
NERatio	0.37	0.33	0.32	0.26	0.27	0.25
NeTokRatio	0.37	0.32	0.32	0.26	0.2760.25	

SimScore	NE	NERatio	NETokRatio
SBERT	0.34	0.34	0.34
InferSent	0.17	0.16	0.16
UniversalSE	0.43	0.43	0.43
SimCSE	0.3	0.3	0.3
LevDist	-0.07	-0.05	-0.05
JaccChar	0.18	0.17	0.17
JaccTok	0.21	0.21	0.21
SeqMat	0.05	0.06	0.07
WordCount	0.37	0.37	0.37
WordRatio	0.32	0.33	0.32
WordTokRatio	0.32	0.32	0.32
SynCount	0.27	0.26	0.26
SynRatio	0.26	0.27	0.27
SynTokRatio	0.25	0.25	0.25
NE	1.0	1.0	1.0
NERatio	1.0	1.0	1.0
NeTokRatio	1.0	1.0	1.0