EEE486 - Natural Language Processing Group-11

Name of The Project: Sentiment Analysis and Semantic Compression of Emails

Group Members:
- Harun Can Surav 21903452 - Övgüm Can Sezen 21902418 CS Department
- Doruk Karaman 21901507 - Berk Özçelik 21901424 EEE Department

Introduction:

Currently, communication systems turn texts into bitwise encoded messages and compress them to send to the destination where they get decoded. However, as the Shannon Limit is being tested with current technology [2], it is essential to further optimize these messages. One idea is to use text summarizers and extenders [1] to compress and expand sent messages. The initial text can be summarized with semantic encoders while still holding its semantic value before going into the usual encryption-decryption process and later be extended with semantic decoders to end up with a text that has the same semantic meaning but with less data transmitted in the process. We believe that a tool like this can be implemented to email services to reduce transmission time and increase the message sent per bit ratio. In order to achieve this goal, we divided the project into 5 main topics which are:

- Semantic Compression
- Sentimental Analysis
- Transportation via byte-wise encoding
- Text Regeneration
- Text Emotion Transfer

In summary, the initial email data will be compressed semantically and a sentimental analysis will be conducted before transporting it in the traditional way to the receiver where the compressed data will be expanded and the sentimental transformation will be done to recover a message similar to the sent one but with less bytes transported.

Semantic Compression:

Text Compression has been going around for decades as explained in the paper by Witten published in 1994. Different types of compression methodologies have been experimented with for many years. Some examples are substituting words with the same meanings as one word which in fact does not shorten the text but make it simpler by reducing the unique word count, this method is called "Thesaurus substitution"; another is removing frequently used but nearly meaningless words such as "the, is, where, …", the words that we describe in the scope of EEE486 lecture as the stop words, from the text file which is called "Lossy Text Compression" this can also be applied by removing vowels from words like writing "how are you" as "hw r y" [3]. The intriguing topic is also related to today's hot subject of AIs. With natural language processing, texts can be shortened to new texts that carry the same semantic meanings.

One of the ways is vector-vise semantic compression which is making a text more generic by using higher order words of that family in place of the original word (see Appendix 1). For example: "The lifespan of a cell depends upon the wear and tear on that cell can be turned into" can be generalized to "The period of time of a cell relies on damage on that cell" [4]. In order to do this process there are two prerequisites. One is removing the stop words, the other is preparing the domain corpora [10]. Ceglarek claims that after the semantic compression the document carries its meaning but in a less complex form. This can be beneficial in classification. The results he got can be seen in the Appendices 2 and 3. It can be seen from the results that even a text that has ¼ complexity of the initial text can still be classified nearly as good. Since our project topic is already limited to business emails, vector wise semantic compression does not have many use cases. The possible use case is having possible cases like condolence mail, congratulation mail and information mail, …etc. and classifying the outgoing mail into the artificial classes. However, this is just a possibility and is not in our current roadmap but it is still in consideration.

With the introduction of the transformer model, which has self-attention, summarization tasks become much easier. Models like BERT (Bidirectional Encoder Representations from Transformers) implemented these transformers to achieve various NLP tasks. In the paper written by Liu [11], methodology for both extractive and abstractive summarization is proposed. For extractive summarization, sentences are first classified whether to be put in the summary Later, the position embeddings of the sentences plus their multi-head attention operation are normalized. Final output is sent to a sigmoid classifier. To train the abstractive summarization module, an encoder-decoder model with two different rated Adam Optimizers [16] is proposed to eliminate the risk of overfitting and underfitting. Our plan is to convert the expression "Can you play some songs by Coldplay?" to something similar to "Intent: PlaySong, Artist: Coldplay", or the expression "Turn off the bedroom light" to "Intent: TurnOffLight, Device: bedroom" [5].

Another use area of semantic compression is managing large data tables. Spartan and ItCompress are two tools that find relations in large SQL files and compress the table by removing linearly dependent rows [6]. This cannot be used for datatables with no error tolerance tables like log-in tables but for tables for user behavior and such with high error tolerance (5%-10%) the tools can compress larger sizes more than the traditional gzip compression. Babu's paper also proves this as the compression increases for Spartan by 2.5-3 times the error tolerance rate which means that Spartan compresses around 14% better than standard gzip when the tolerance is about 5%. When comparing Spartan to ItCompress can perform better as it is the successor to Spartan with a ratio of 1.5 to 3 times better compression [7]. However, data table compression is not planned to be used in the project but was researched to get a general idea about possible tools.

Re-pair is a tool which is used for pairing high frequency words or word pairs to grammatical symbols, thus making the initial text smaller in size. Basically, if X is a new symbol not appearing in S which is a string (for algorithm see Appendix 4). The rule X → ab is added and all occurrences of ab in S are replaced with X [8]. Which implied that the ab is now paired with X. Bille claims that despite its simple approach to the problem, "Re-Pair achieves high-order entropy compression and—especially on repetitive datasets—is an excellent compressor" (for results see Appendix 5). This tool can be paired with the vector generalized text to compress it even more which may be a solution to some future problems but as mentioned under the vector-vise compression, this methodology is not our number one approach to the problem.

All the aforementioned tools and techniques can be implemented in our project and can be used to improve it; however, the approach we wanted to pursue before bitwise encoding the mail would be extracting information from the mail text into a table. Information extraction means that the algorithm identifies necessary information from the text file in our case the business mail and stores the information on a table. This information can be a meeting place, time, attendees, name of an important event, etc. The model developed by Wu, Zhang and Li is a sec2sec model which means it requires a sequential input, which in our case is words, and outputs a sequence of items but is different from vanilla sec2sec with "table constraint and table relation embeddings". The model is a BART based model which uses a transformer. The results of the tool can be seen in the Appendix 6 [9]. The proposed model is similar to what we want to use to compress the mail text by only storing the important data entries in the mail and sending it to the receiver.

Sentiment Analysis:

Sentiment analysis is the technique which deals with feelings, feedbacks, emotions along with perceptions, which is produced from data, being considerably used in fields like social media analysis since sentiments are one of the most crucial properties to determine the person's behavior [17]. The sentiment of a given sentence can be classified in polarities -as positive, negative, neutral etc.-. However, a polarity also includes many emotions, such as positive including happiness or affection, while a negative polarity can include disgust or sadness [18]. Thus, a block of text built of such sentences would then have multiple polarities and/or emotions with varying amounts of intensity (for example very angry and a little sad). Since the encoded messages will be attempted to be reconstructed according to their sentimental values, it is very important to use more accurate definitions instead of broader umbrella terms to get as close as possible to the pre-encoded message. In order to be able to transfer these varying amounts of emotions, the intensities of those emotions can be expressed with coefficients/weights according to their abundance, and then those weights and polarities would also be transmitted with the encoded sentences.

While tackling this problem, there are two main methods: rule/lexicon based or machine learning based [19]. In lexicon based methods, each word is processed separately, and it is assumed that the sentimental value of the sentence is determined according to the majority. The main problem with this method is that the values of words are not determined contextually, thus could end up with inaccurate results. An example for this could be the word “small” in “small chance for error” has a positive implication while “small chance for success” has a negative one. With the added fact that this method is efficient in separating polarities but not as efficient in separating a polarity into its smaller subset of emotions since it fails in separating ambiguities due to its binary nature (a word being positive, negative or neutral is considered) [20]. Because of all these mentioned aspects of lexicon based methods, machine learning based methods will be utilized in this project.

When different methods such as BERT, XLM-R, ELMO and Zero Shot Classification (ZSC) are looked into, the ones that are more applicable for our project are BERT and ZSC [17], [18], [20]. Compared with one another, the advantages of using ZSC are much more beneficial for this project. First of all, while BERT requires training data, ZSC doesn’t. This is beneficial in the way that if we were to expand our scope to mails in general, rather than business mails, our data set only consisting of business mails would not be a limiting factor since ZSC does not require training data. Also, to be able to accurately function in our project, BERT requires domain specific fine-tuning [17], [21]. However, ZSC detects emotions first, which then uses them to detect the sentimental polarity of the sentence, which is very convenient for our project since we are trying to detect the emotions. Another benefit of ZSC is that there is no need for complicated feature extraction or ML methods, since it does not need the sentiment and rather deduces the sentiment; which accelerates the analysis process. This is also convenient for this project since in essence, our main goal is optimizing mail transmissions.

The solution for semantic analysis discussed in Tesfagergish et al. [18] proposes a two stage solution where the first stage is an unsupervised zero-shot learning model based on a sentence transformer, which would be returning the probabilities for subsets of 34 emotions (the list of emotions is in Appendix 7). Then the output of this stage is used to train the machine learning classifier which would be utilized to determine the polarity of a given text. The classifier is trained on the sentiment labels in a supervised manner using ensemble learning to fulfill this task. For our purposes, the second layer can be changed so that it also returns emotions instead of polarity, since polarity is not important (or at least less important for our means); or can be fully discarded depending on the results.

Transportation via byte-wise encoding:

Since the main goal of this project is optimizing the process of mailing at a technical level, the intercommunication aspect of mailing should also be considered. With that being said, this aspect of communication is not really related with NLP, and more so related with topics such as communication, telecommunication or digital signal processing. So, any methodology with sufficient amount of compression is applicable for this project. With the added fact that the results will be tested on ideal conditions (no telecommunication, both sender and receiver shall be the same device), the probability related parts and error compensation algorithms that are required for telephony are essentially redundant. As a result, any algorithm with sufficient compression capabilities would be sufficient for now (In a case where this project is decided to be continued to produce a marketable product, more research can be conducted to further optimize this intermediary part). For the type of algorithm that will be used (at least for now), we decided on implementing a byte-pair encoding method on bit level (in addition to the previous similar text level encodings)[8]. The reasoning for this being that even though it is computation wise very expensive, the output is sufficiently compressed [22]. It also grants us the possibility of reinforcing something related with our course in a part that is not related with NLP.

Text Regeneration:

As the semantic data is transported back to the receiver, the next step is regenerating a text that has the same semantic meaning and sentiment. We found two viable approaches that achieve the goal of retaining both sentiment and semantic meaning. First is to generate the text using only semantic data and then applying the sentiment data. The latter option is to generate the text with both semantic meaning and sentiment data in mind.

We found several systems that generate texts from semantic data; but as we are still not set on what type of data to transfer (tabular, structured or semi structured sentence), our survey report contains alternatives for both cases.

Harkous et al. [12] proposes DATATUNER; a neural, end-to-end system that achieves text expansion from tabular (structured) data that is not domain specific. They propose a 2-stage architecture which first generates and re-ranks. Generator is a fine-tuned model that is built on GPT-2 [15] architecture. Upon GPT-2, Harkous et al. [12] added state embeddings that help the model distinguish the type of data that is being handled. To train the generator; input embeddings, positional embeddings and state embeddings are summed and fed to the GPT-2 layer. Last layer of GPT-2 is normalized and gets tied to the input embeddings. Finally a softmax is applied to get the probability distribution. After the generation task is completed, outputs are fed into a Semantic Fidelity Classifier which determines whether the generated text is accurate or contains some errors (hallucination, omission, repetition or value error). Ranker (decoder) is an algorithm that implements beam-search. Ranking key is the product of conditional probabilities multiplied with a length normalization factor. Low scoring candidates are dropped after beam length is breached. In their experiments, they outperformed the state of the art in automated tests (see Appendix 8).

In Data-to-Text Generation with Content Selection and Planning, Puduppully et al. [13] proposes an end-to-end neural network architecture that achieves text expansion from tabular (structured) data with considering content selection and content planning as separate tasks. In order to prepare the data for the content selector module (content selection gate), tabular data is first transformed through a record encoder which transforms the data into a vector through a rectified linear unit (ReLU). Content selection gate computes the attention scores over the input to determine the significant features that should be in the generated text. For our purposes, content selection gate will be mostly neglected as the transferred data only contains semantically important data. In their paper, Puduppully et al. [13] points out the importance of order of sentences in an overall text. To create a content plan, inputs to the content planning module are denoted to pointers and probabilities of sequences are calculated. Pointer networks [14] are used for feeding the pointers into LSTM that generates the positioning tokens. Generation of text is achieved by giving the generating recurrent neural network (consisting of bi-directional LSTMs) the input vectors and conditioning it over the content planner. Results of their finding shows that the proposed architecture has an advantage over state of the art systems of 2019 (see Appendix 9).

When compared, DATATUNER outperforms the latter architecture in BLEU scores by a significant margin (see Appendices 7 and 8). The fact that it is generalizable is also useful for our domain. In addition, use of pre-trained models decreases training time during the implementation stage, which is another win for DATATUNER.

Text Emotion Transfer:

Another important aspect of regenerating the text is making sure that it implies the same emotional tone (style) of the original text as it is an important part of communication in natural languages. To achieve this we decided to add one more generative step on top of the previously explained generation of text. This second step of generation needs to conserve the semantic and syntactic characteristics of the input and edit it to make it reflect the wanted emotional style.

This process in general is called Text Emotion Transfer (TET), which is similar to Text Sentiment Transfer (TST), but instead of transforming text into “positive” and “negative” (or any value between them), it aims to transform it into predefined emotions like “anger”, “sadness”, “joy”. [23] TET is a Natural Language Generation (NLG) process as the main approach is to create a new text –from an original text– with a predefined emotional style in combination of the semantic and syntactic characteristics of the original text. [23] More formally this problem can be defined as:

$$TET(t,\ e') = argmax_{t'}\, P_{c,e'}(t' \mid t,\ e')$$

Where TET(t, e’) stands for the generation function with t and e’ as inputs, t stands for the original text, t’ stands for the generated text and e’ stands for the target emotion. [23]

The typical implementation of TET systems –similar to TST systems– is to follow a rule-based approach which first masks the emotion-attributed lexical tokens and then fills their spaces with tokens belonging to the target emotional style, generally using sequence-to-sequence (seq2seq) models. [23] The most important (and common) difficulty of this approach is finding an emotion-labeled large corpus. To overcome this issue, it is common to follow unsupervised learning models like “style content disentanglement” using approaches like adversarial learning. [23]

To find a solution to be able to implement on our proposed system, four models from four papers [24] [25] [26] [27] are investigated. The methodology followed to select a possible solution amongst the five papers was to first evaluate the results. The important quality we are looking for in our project is that while generating the new text, the grammar structure and the semantic characteristic of the previous text is conserved while the target emotion is reflected successfully.

The "experiment" parts of the papers are read to assess how much the proposed models satisfy the expectations from our system described above. For a qualitative evaluation, we expected a metric to show how successful the model was at generating the new text to reflect the target emotion (in the future we will refer to it as the emotion metric) and another to show how similar it is to the original text in terms of its content (in the future we will refer to it as the content metric). [23] Also, some examples are expected –as input output pairs– to see if the proposed model causes any grammar or syntax errors in the generated sentences. There is a slight dependency on human evaluation in these models as it is seen as a good idea to evaluate the generated sentences by real people. [24]

The automatic evaluation metrics in papers [25] [26] [27] used BLEU as the evaluation metric for the models ability to preserve the input text content and an "Accuracy" metric acquired from the fastText classifier –a model which classifies the polarity of the text input– tuned for the emotion of interest. [26] The other paper of interest [24] has self-defined metrics for the same two evaluation concepts. Also, papers [24] [27] include some examples of input-output pairs of sentences. Paper [23] does not propose a specific model, instead it focuses on explaining common issues and concepts together with experimenting on several previously proposed models.

Upon investigation, the models proposed in papers "SentiInc: Incorporating sentiment information into sentiment transfer without parallel data*"* and *"Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer"* are eliminated due to the experiments result being less convincing as the information inside the paper is limited to that of the other papers. [26] [27] Also, the two papers focus on STS and require a large-scale adaptation to be able to use them for our purpose which is not justified considering the experiment result information being less convincing in comparison to other proposed models in papers [24] [25]. The remaining two solutions will be experimented with to see which of those provide the better solution for our system.

The model proposed by paper *"Generative Sentiment Transfer via Adaptive Masking"* is named AM-ST. AM-ST considers the tokens to be masked as learnable variables and uses another model to generate the Bi-LSTM model to generate the hidden vectors. Later these hidden vectors and an attention weight vector are used to get the probability of a word to be in need of masking. [25] AM-ST also incorporates two classification losses, "Sentiment Classification Loss" and "Content Classification Loss". [25] Sentiment Classification Loss employs a pre-trained classifier which accepts the set of tokens masked to measure the set of tokens to be masked is associated with the emotion information. [25] Content Classification Loss employs a BoW distribution to measure the closeness of the non-masked tokens in the text with the original text. [25] AM-ST also employs adversarial losses to ensure that the set of non-masked tokens have as little sentiment information as possible and the set of tokens masked to contain as little content information as possible. [25] AM-ST uses sentiment-aware masked language model (Senti-ML) to fill the places of the masked tokens in the text, using the loss definitions to fine-tune it. [25]

The model proposed by paper *"Emotional text generation based on cross-domain sentiment transfer."* utilizes a sequence-to-sequence (Seq2Seq) based NN to generate the text. [24] The model also employs a discriminator –RNN-based classifiers– to distinguish if a sentence is from the generator and or an original one. [24] Then, an adversarial reinforcement learning approach is used, training the generator to beat the discriminator. [24] Loss functions similar to that of the AM-ST model without the adversarial losses to fine-tune the Seq2Seq model. [24]

The two models will be experimented, while being implemented on our own system to see which one fits better.

References:

[1] H. Xie and Z. Qin, “A Lite Distributed Semantic Communication System for Internet of Things,” IEEE JSAC, vol. 39, no. 1, Jan. 2021, pp. 142–53.

[2] H. Xie, Z. Qin, G. Y. Li and B. -H. Juang, "Deep Learning Enabled Semantic Communication Systems," in IEEE Transactions on Signal Processing, vol. 69, pp. 2663-2675, 2021

[3] I. H. Witten, T. C. Bell, A. Moffat, C. G. Nevill-Manning, T. C. Smith, and H. Thimbleby, “Semantic and Generative Models for Lossy Text Compression,” *The Computer Journal*, vol. 37, no. 2, pp. 83–87, Jan. 1994, doi: https://doi.org/10.1093/comjnl/37.2.83.

[4] D. Ceglarek,. "Semantic compression for text document processing*" Transactions on Computational Collective Intelligence,* vol XIV (2014): 20-48.

[5] P. Prakash, S. K. Shashidhar, W. Zhao, S. Rongali, H. Khan, and M. Kayser, “Compressing transformer-based semantic parsing models using compositional code embeddings,” *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[6] S. Babu, M. Garofalakis, and R. Rastogi. "Spartan: A model-based semantic compression system for massive data tables." *ACM SIGMOD* Record 30.2 (2001): 283-294.

[7] H. V. Jagadish, R. T. Ng, Beng Chin Ooi and A. K. H. Tung, "ItCompress: an iterative semantic compression algorithm," *Proceedings. 20th International Conference on Data Engineering,* Boston, MA, USA, 2004, pp. 646-657, doi: 10.1109/ICDE.2004.1320034.

[8] P. Bille, IL. Gørtz, and N. Prezza. "Practical and effective re-pair compression" *arXiv preprint* arXiv:1704.08558 (2017).

[9] X. Wu, J. Zhang, and H. Li,”Text-to-Table: A New Way of Information Extraction” vol. 1, pp. 2518–2533, 2022, Available: https://aclanthology.org/2022.acl-long.180.pdf

[10] D. Ceglarek,. "Semantic Compression for Specialized Information Retrieval Systems*" Advances in Intelligent Information and Database Systems* (pp.111-121).

[11] Y. Liu and M. Lapata, ‘Text Summarization with Pretrained Encoders’, *arXiv [cs.CL]*. 2019.

[12] H. Harkous, I. Groves, and A. Saffari, ‘Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity’, *arXiv [cs.CL]*. 2020.

[13] R. Puduppully, L. Dong, and M. Lapata, ‘Data-to-Text Generation with Content Selection and Planning’, *arXiv [cs.CL]*. 2019.

[14] O. Vinyals, M. Fortunato, and N. Jaitly, ‘Pointer Networks’, *arXiv [stat.ML]*. 2017.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, ‘Language Models are Unsupervised Multitask Learners’, 2018.

[16] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, *arXiv [cs.LG]*. 2017.

[17] K. Pipalia, R. Bhadja and M. Shukla, "Comparative Analysis of Different Transformer Based Architectures Used in Sentiment Analysis," *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India, 2020, pp. 411-415, doi: 10.1109/SMART50582.2020.9337081.

[18] S. G. Tesfagergish, J. Kapočiūtė-Dzikienė, and R. Damaševičius, ‘Zero-Shot Emotion Detection for Semi-Supervised Sentiment Analysis Using Sentence Transformers and Ensemble Learning’, *Applied Sciences*, vol. 12, no. 17, 2022.

[19] W. Medhat, A. Hassan, and H. Korashy, ‘Sentiment analysis algorithms and applications: A survey’, *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[20] M. Wankhade, A. C. S. Rao, and C. Kulkarni, ‘A survey on sentiment analysis methods, applications, and challenges’, *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5731–5780, Oct. 2022.

[21] C. Sun, X. Qiu, Y. Xu, and X. Huang, ‘How to Fine-Tune BERT for Text Classification?’, in *Chinese Computational Linguistics*, 2019, pp. 194–206.

[22] K. Bostrom and G. Durrett, ‘Byte Pair Encoding is Suboptimal for Language Model Pretraining’, *arXiv [cs.CL]*. 2020.

[23] R. MohammadiBaghmolaei and A. Ahmadi, “TET: Text emotion transfer,” Knowledge-Based Systems, vol. 262. Elsevier BV, p. 110236, Feb. 2023. doi: 10.1016/j.knosys.2022.110236.

[24] R. Zhang, Z. Wang, K. Yin, and Z. Huang, “Emotional Text Generation Based on Cross-Domain Sentiment Transfer,” IEEE Access, vol. 7. Institute of Electrical and Electronics Engineers (IEEE), pp. 100081–100089, 2019. doi: 10.1109/access.2019.2931036.

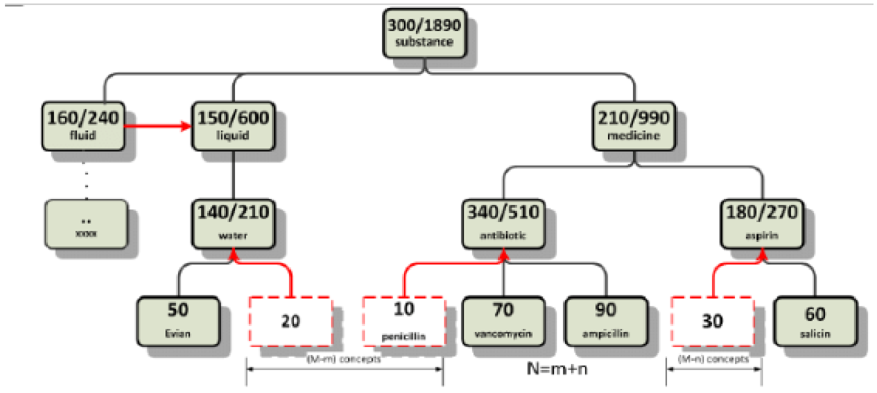
[25] Y. Xie, J. Xu, L. Qiao, Y. Liu, F. Huang, and C. Li, "Generative Sentiment Transfer via Adaptive Masking." arXiv, 2023. doi: 10.48550/ARXIV.2302.12045.

[26] K. Pant, Y. Verma, and R. Mamidi, "SentiInc: Incorporating Sentiment Information into Sentiment Transfer Without Parallel Data," Lecture Notes in Computer Science. Springer International Publishing, pp. 312–319, 2020. doi: 10.1007/978-3-030-45442-5_39.

[27] Li, R. Jia, H. He, and P. Liang, "Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer." arXiv, 2018. doi: 10.48550/ARXIV.1804.06437.

Appendices:

Appendix 1: Generalizing Words with vectors



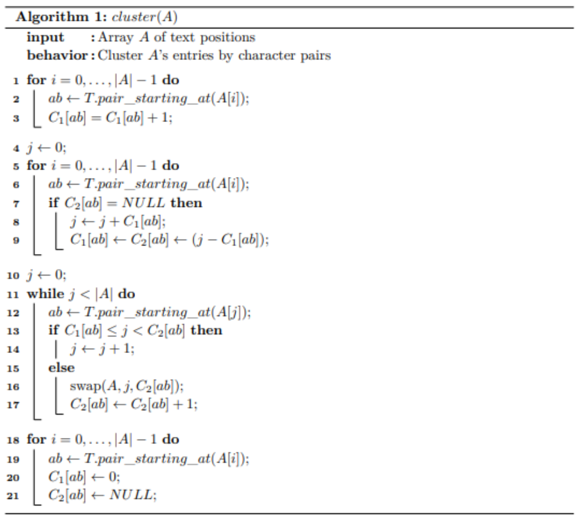
Appendix 2: Vector-vise semantic compression results 1

Number of clustering features	1000	900	800	700	600	Average
Without semantic compression	93,46%	90,90%	91,92%	92,69%	89,49%	91,69%
12000 concepts	91,92%	90,38%	90,77%	88,59%	87,95%	89,92%
10000 concepts	93,08%	89,62%	91,67%	90,51%	90,90%	91,15%
8000 concepts	92,05%	92,69%	90,51%	91,03%	89,23%	91,10%
6000 concepts	91,79%	90,77%	90,90%	89,74%	91,03%	90,85%
4000 concepts	88,33%	89,62%	87,69%	86,79%	86,92%	87,87%
2000 concepts	86,54%	87,18%	85,77%	85,13%	84,74%	85,87%
1000 concepts	83,85%	84,10%	81,92%	81,28%	80,51%	82,33%

Appendix 3: Vector-vise semantic compression results 2

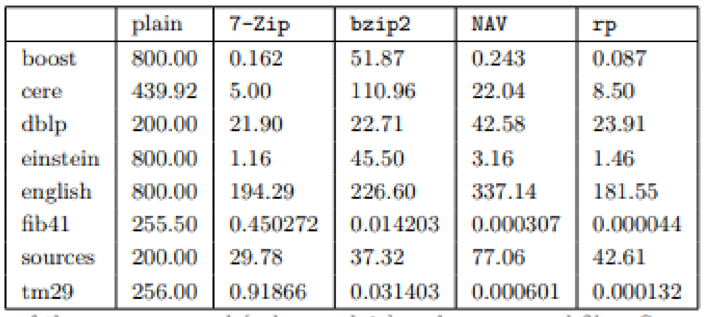
Number of clustering features	1000	900	800	700	600	Average
Without semantic compression	93,78%	93,89%	93,11%	92,56%	92,11%	92,03%
12000 concepts	93,00%	94,00%	94,00%	91,33%	90,78%	91,49%
10000 concepts	93,33%	94,22%	93,56%	93,44%	92,22%	92,33%
8000 concepts	92,78%	93,22%	94,22%	93,33%	90,89%	91,79%
6000 concepts	92,56%	93,44%	92,22%	92,89%	91,00%	91,26%
4000 concepts	92,00%	92,44%	91,22%	90,89%	90,22%	90,03%
2000 concepts	92,33%	91,78%	89,89%	90,56%	89,67%	89,44%
1000 concepts	92,00%	92,00%	88,33%	87,11%	83,78%	86,90%

Appendix 4: re-pair algorithm



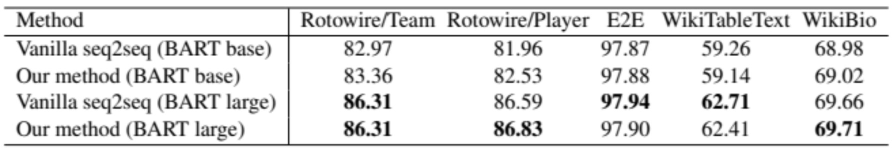
```
Algorithm 1: cluster($A$)
  input    : Array $A$ of text positions
  behavior : Cluster $A$'s entries by character pairs
for $i = 0, \dots, |A| - 1$ do
    $ab \leftarrow T.pair\_starting\_at(A[i])$;
    $C_1[ab] = C_1[ab] + 1$;
$j \leftarrow 0$;
for $i = 0, \dots, |A| - 1$ do
    $ab \leftarrow T.pair\_starting\_at(A[i])$;
    if $C_2[ab] = NULL$ then
        $j \leftarrow j + C_1[ab]$;
        $C_1[ab] \leftarrow C_2[ab] \leftarrow (j - C_1[ab])$;
$j \leftarrow 0$;
while $j < |A|$ do
    $ab \leftarrow T.pair\_starting\_at(A[j])$;
    if $C_1[ab] \leq j < C_2[ab]$ then
        $j \leftarrow j + 1$;
    else
        swap($A, j, C_2[ab]$);
        $C_2[ab] \leftarrow C_2[ab] + 1$;
for $i = 0, \dots, |A| - 1$ do
    $ab \leftarrow T.pair\_starting\_at(A[i])$;
    $C_1[ab] \leftarrow 0$;
    $C_2[ab] \leftarrow NULL$;
```

Appendix 5: re-pair document size comparison



	plain	7-Zip	bzip2	NAV	rp
boost	800.00	0.162	51.87	0.243	0.087
cere	439.92	5.00	110.96	22.04	8.50
dblp	200.00	21.90	22.71	42.58	23.91
einstein	800.00	1.16	45.50	3.16	1.46
english	800.00	194.29	226.60	337.14	181.55
fib41	255.50	0.450272	0.014203	0.000307	0.000044
sources	200.00	29.78	37.32	77.06	42.61
tm29	256.00	0.91866	0.031403	0.000601	0.000132

Appendix 6: Results of text to table information extraction

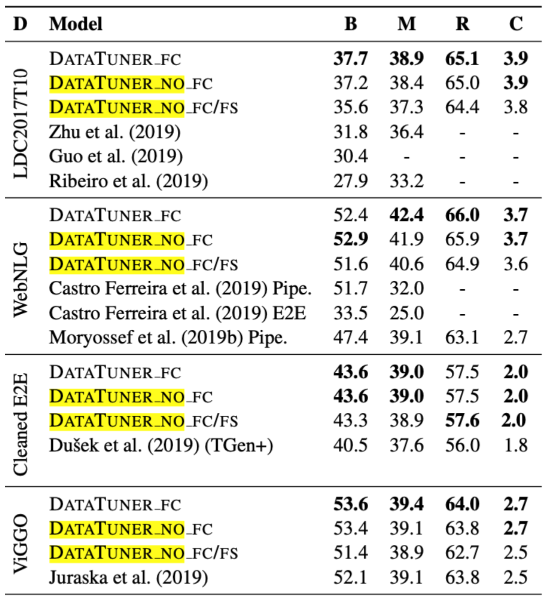


Method	Rotowire/Team	Rotowire/Player	E2E	WikiTableText	WikiBio
Vanilla seq2seq (BART base)	82.97	81.96	97.87	59.26	68.98
Our method (BART base)	83.36	82.53	97.88	59.14	69.02
Vanilla seq2seq (BART large)	**86.31**	86.59	**97.94**	**62.71**	69.66
Our method (BART large)	**86.31**	**86.83**	97.90	62.41	**69.71**

Appendix 7: List of subsets of emotions for the Zero Shot method

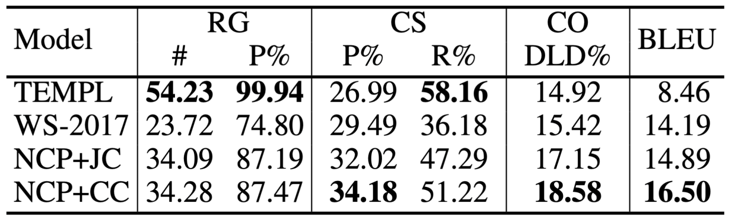
Anger, sadness, disgust, fear, joy, happiness, admiration, affection, anguish, caution, confusion, desire, disappointment, attraction, envy, excitement, grief, hope, horror, joy, love, loneliness, pleasure, fear, generosity, rage, relief, satisfaction, sorrow, wonder, sympathy, shame, terror, and panic.

Appendix 8: DATATUNER vs. state of art system. (D: dataset, B: BLEU score, M: METEOR score, R: ROUGEL score, C: CIDEr score.)



D	Model	B	M	R	C
LDC2017T10	DATATUNER_FC	**37.7**	**38.9**	**65.1**	**3.9**
	DATATUNER_NO_FC	37.2	38.4	65.0	**3.9**
	DATATUNER_NO_FC/FS	35.6	37.3	64.4	3.8
	Zhu et al. (2019)	31.8	36.4	-	-
	Guo et al. (2019)	30.4	-	-	-
	Ribeiro et al. (2019)	27.9	33.2	-	-
WebNLG	DATATUNER_FC	52.4	**42.4**	**66.0**	**3.7**
	DATATUNER_NO_FC	**52.9**	41.9	65.9	**3.7**
	DATATUNER_NO_FC/FS	51.6	40.6	64.9	3.6
	Castro Ferreira et al. (2019) Pipe.	51.7	32.0	-	-
	Castro Ferreira et al. (2019) E2E	33.5	25.0	-	-
	Moryossef et al. (2019b) Pipe.	47.4	39.1	63.1	2.7
Cleaned E2E	DATATUNER_FC	**43.6**	**39.0**	57.5	**2.0**
	DATATUNER_NO_FC	**43.6**	**39.0**	57.5	**2.0**
	DATATUNER_NO_FC/FS	43.3	38.9	**57.6**	**2.0**
	Dušek et al. (2019) (TGen+)	40.5	37.6	56.0	1.8
ViGGO	DATATUNER_FC	**53.6**	**39.4**	**64.0**	**2.7**
	DATATUNER_NO_FC	53.4	39.1	63.8	**2.7**
	DATATUNER_NO_FC/FS	51.4	38.9	62.7	2.5
	Juraska et al. (2019)	52.1	39.1	63.8	2.5

Appendix 9: NCP (proposed system) vs. others with description.



Model	RG		CS		CO	BLEU
	#	P%	P%	R%	DLD%	
TEMPL	**54.23**	**99.94**	26.99	**58.16**	14.92	8.46
WS-2017	23.72	74.80	29.49	36.18	15.42	14.19
NCP+JC	34.09	87.19	32.02	47.29	17.15	14.89
NCP+CC	34.28	87.47	**34.18**	51.22	**18.58**	**16.50**

Table 5: Automatic evaluation on ROTOWIRE test set using relation generation (RG) count (#) and precision (P%), content selection (CS) precision (R%) and recall (R%), content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%), and BLEU.