Applicability of Language Models to Fact Checking

George Karagiannis Florian Suri-Payer Himank Yadav

Abstract

Modern information platforms are becoming increasingly crowd-sourced and editable by the general public (e.g. Wikipedia) and are consequently easily affected by large quantities of false statements. The sheer volume of information makes it near impossible for humans to regulate, and thus it is becoming important to automatically identify inconsistent and factually false information. However traditional fact checking requires access to large, verified information stores which is both slow and expensive. Since language models are trained on such data sets they potentially implicitly learn factual contexts that may be leveraged for *passive* automated fact-checking. We investigate the use of a Language Model based approach to fact-checking that is based on computing a heuristic likelihood of individual facts. We explore 3 state-of-the-art Language Models across different threshold models and evaluate the precision and recall over a small fact update data-set.

1. Motivation

Modern online news and information platforms are increasingly accessible to and editable by the broader public. Open, and mutable information sources are easily permeated by large quantities of false statements, compromising the integrity said source. Given the enormous volume of information generated online it is nearly impossible for traditional fact checking methods (human-based, i.e. Journalists) to keep up. Thus, it is naturally desirable to develop and employ computational fact-checking tools that aim to identify and correct inconsistent or factually false information. Fact checking and Fake News identification are important emerging research areas that focus on maintaining veracity when consulting available information on the web.

A cardinal example for open, mutable information sources is Wikipedia, "a free online encyclopedia, created and edited by volunteers around the world" that verified purely on a peer-to-peer basis and automatically checked for its content accuracy. While Wikipedia is nowadays considered fairly thrust-worthy, due to its large reach and resulting participation, factual mistakes on the Web are generally not avoidable and frequently result in frustration using the sources. There are two fundamental approaches to consider when developing automated fact checking methods. An automated tool may follow an *active* fact-checking approach, i.e. attempt to actively search for incorrect facts and either just flag them, or ideally replace them with correct facts. Inserting or predicting a true fact however, is vastly more difficult than ruling out improbable facts and generally requires a large knowledge-base of true information. A different approach is to *passively* monitor inserted/updated information and regulate the introduction of false information in the first place.

In order to perform any form of fact-checking it is necessary to have a reference of truth. Thus it is required to either have access to a knowledge-base of verified true facts in order to exclude and replace mistakes or a collection of known false facts, i.e. an anti-knowledge-base, in order to just exclude facts. However, creating, maintaining and searching such knowledge-bases is both difficult and expensive. This raises the question, whether it is possible to learn facts not as collection, but instead train a compact model that is capable of predicting facts and their likelihood. A loosely related string of research that fits this approach description are Language Models. Language Models aim to solve a similar problem, where instead of facts models try to predict or assess sentence construction. These models learn to model natural language by training on large quantities of text data that serves as correct reference. Interestingly however, modern state-of-the-art language models are learned nearly entirely over book corpora and Wikipedia. If we assume that these sources are largely factually accurate (Wikipedia certainly contains mistakes, but if we assume that it is mostly correct and mistakes do not appear across duplicates in different articles this can be amortized) then these Language Models were in fact trained on a factual knowledge-base. Since modern Language Models learn dependencies through deep contextual relationships, this raises the question whether they implicitly acquired a contextual model for facts during training.

In the following we investigate this question by employing and analyzing different state-of-the-art Language Models in a passive fact-checking approach.

2. Related Work

The domain of automated fact-checking is currently sparsely researched. Most of current work associated with Fact Checking makes use of Knowledge Bases or Knowledge Graphs to explore the whether target statements are true or false. Vlachos & Riedel (2014) construct a labeled dataset from data extracted by Politifact that are hand-classified as Factual Mistakes. They propose supervised learning techniques in order to learn factual classification. Ciampaglia et al. (2015) approach the fact-checking task by investigating correspondence between shortest paths in knowledge graphs that represent factual relations. Another Fact Checking System, ClaimBuster (Hassan et al., 2017) builds a model using a human-labeled dataset and employs several supervised learning methods such as Multinomial Naive Bayes Classifier (NBC), Support Vector Machine (SVM) and Random Forest Classifier (RFC). All of these approaches use dedicated Knowledge Bases as external structured information for either output or learning.

Language in context of fact-checking has been previously explored by Rashkin et al. (2017), however their approach focuses on discerning facts such as political statements based on their form of expression.

3. Fact-Checking Architecture

3.1. Update validation stream

Our work aims to investigate an unstructured probabilistic setting by employing Language Models in order to surmise whether statements are factual. Thus, we target a heuristic approach to fact-checking rather than confirming actual truth. An initial step to fact-checking using Language Models is to focus on the recognition of the factual mistakes, rather than their rectification. We attempt to leverage Language Models for a passive fact-checking domain such as an update validation heuristic to some information store (i.e. Wikipedia). The goal of a passively monitoring update procedure is to increase the overall truth value of the information store. In order to do so, the heuristic should allow incorrect facts to be replaced while prohibiting correct facts to be removed. The heuristic may be indifferent to updates form wrong to wrong facts as this does not change the truth value of the information store.

In order to judge the veracity of a fact we use a Language Model to predict a facts likelihood given a context. Intuitively, a token with low predicted likelihood may be considered false, as it is unlikely to be consistent with the context. Thus, given two or more choice and some likelihood distribution (based of a Language Model) we may rule out improbable facts. This avoids making a definitive decision about the true solution, while following the fact-checking intuition that an unlikely choice (given alternatives) is presumably false.

Specifically we model an update stream as sequence of sentences $U = S_1, S_2, \ldots, S_t$. Each sentence is comprised of a single token, denoted F for fact, and its composing context, denotes C. Thus a sentence can be modelled as $S = C\{F/\_\}$, i.e. the substitution of a fact F into some blank inside the context C. In the example sentence *"Germany won the world cup in 2014"* the fact could be $F = Germany$ and the context $C =$ *"_ won the world cup in 2014"*. Depending on the Language Model, the context may be ordered or un-ordered and the position of F may or may not be considered (see section 4). Moreover, any two adjacent sentences in the update stream differ only in one token, i.e. the fact: $S_2 = S_1\{F_2/F_1\}$ (and therefore $C_1 = C_2$).

In order to compare sentences and validate updates we must compute the likelihood of a proposed fact F given some context C: $P(F|C)$. We calculate this likelihood using a Language Model as described in Section 4. Note that the "likelihoods" computed by our models are not normalized (i.e. $\sum P(F|C) > 1$) and technically are not probabilities (rather a similarity metric), yet correspond to the same relationship.

3.2. Static Threshold Model

Given a current state S_1 of the store, and some proposed updated state S_2 the fact-checker validates whether this update should be legal by comparing the likelihoods of both states. In our future analysis we treat allowed updates from a factually incorrect to a correct state as True Positive and allowed updates from correct states to incorrect ones as False Positives. As previously mentioned, the decision for updates that do not affect the truth value may be arbitrary.

An update is allowed if:

$$TM(S_1, S_2) == true$$

Where $TM()$ is a Threshold model formula that specifies how much more likely the updated state must be. Explicitly we initially consider 2 basic Threshold models, a *relative* and an *absolute* Threshold respectively. As the denotations suggest, the relative Threshold model compares the relative differences between likelihoods:

$$relTM \triangleq \frac{P(F_2|C)}{P(F_1|C)} > T$$

This model imposes a tune-able comparative Threshold $T > 1$ on the update, demanding that the updated state must be T times more likely in order gauge the confidence that the update will not replace a correct fact.

Similarly, the absolute Threshold models requires the updated state to be a constant percentage T more likely.

$$absTM \triangleq P(F_2|C) - P(F_1|C) > T$$

An absolute threshold not only imposes an increase in likelihood but also a minimal likelihood that the update needs to satisfy.

The absolute Threshold intuitively reduces both False positives (true to false) and neutral updates (false to false) as those are states are supposedly of low likelihood. It requires at least the threshold of certainty in the updated state and thus attempts to avoid allowing updates, even when the updated state is more likely. The cost of doing so is also incurring more False negatives, as potentially correct updates are deemed unlikely. In comparison, the relative Threshold may allow updates even if both states have very low likelihood. Instead, the relative Threshold may be more conservative when both options have high likelihood. Effectively this makes updates harder the more confident we are in the current state. Intuitively this corresponds to the "stakes" of updating being higher, because the relative gain in truth value is lower. This supposedly limits false positives: When both sentences have low likelihood we could interpret it as neutral update (false to false), whereas when both sentences are of high likelihood, one of them is more likely correct and it is critical to avoid a mistake (true to false). While stricter, this notion contradicts the intuition that two highly probable "facts" may correspond to qualitatively (i.e. Berlin vs Munich being the capital of Germany) similar truth values. We address this issue in section 6. A mixture of both Threshold models allows for extended fine-tuning.

3.3. Fact prediction

In an ideal scenario, Language Model methods methods would be capable of not only identifying, but also replacing inconsistent facts. However, Masked Language Models traditionally aim to solve the task of completing words in order to fit a sentence structure and multiple admissible solutions may exists. The choice that is usually committed to is the most probable word over a continuous distribution. In the fact checking application there is only one correct solution and all other answers are inherently incorrect. While one may try to predict correct facts by choosing the most likely token for any given context, this is coupled with uncertainty, as a lot of supposed "facts" may be close in likelihood. This paradigm rift between continuous losses for the traditional language setting and a 0/1 loss for the fact checking domain makes predicting a true fact risky in settings with large variance of possible answers. An active fact checking domain should therefore be avoided.

In the following we expand on the employed Language Models and how to derive the conditional likelihoods.

4. Language Model Background

We use two different approaches to predict whether the updated fact is more likely that the original one given the context of the sentence: Masked Language Models and word embeddings.

The first approach uses a Masked Language model that models $P(F|C)$ directly from a sentence $S = F \cup C$. For example for the sentence $S =$ *"France won the world cup in 2018"*, where $F = France$ and $C = S \setminus F$, the Masked Language model will approximate $P(France|C)$. As our Language model, we used the state of the art model, BERT (Devlin et al., 2018), short for Bidirectional Encoder Representations from Transformers. BERT makes use of the Transformer model (Vaswani et al., 2017), in order to learn bidirectional contextual relations between words in a text corpus. To do that, BERT uses two pre-training strategies.

1. Masked Language Model (MLM): Hide 15% of the input tokens at random and try to predict only the masked ones. The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard Language Model.

2. Next Sentence Prediction (NSL): To understand the relationship between sentences, BERT pre-trains a next sentence prediction task. Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus.

Unlike other Transformer Language Models, BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers.
MLM is directly applicable to our setting, whereas NSL is more applicable to downstream NLP tasks such as Question Answering and Natural Language Inference. Given a sentence with masked token, i.e. C and a candidate token, i.e. F, MLM outputs a logit corresponding to the "suitability" of the token. We transform the logit into a likelihood $P(F|C)$.

The second method we employ concerns the use of word embeddings that are derived from three different models: GloVe (Pennington et al., 2014), ELMo (Peters et al., 2018) and BERT. For each token in a sentence,

the respective model computes a vector capturing its semantic representation. The general idea behind using word embeddings is to use the semantic representation for each word in a sentence, in order to derive a semantic representation for the context C and the fact F. The end goal is to approximate $P(F|C)$, using those embeddings.

We use multiple models that learn different vector representations and use the produced word embeddings to compare their performance.

1. GloVe: The GloVe emdeddigs that we use in this project were trained on CommonCrawl and contain over 1 million unique vectors. Given a sentence $s = [t_1, t_2, ..., t_n]$ and a target token t_t, GloVe vectors are obtained using two different methods:
   (a) trying to predict the neighbor tokens $[t_i, .., t_{i+w}]$ in a certain window w given a target token t_t in a sentence. This approach is called the Continuous Bag of Words Model.
   (b) trying to predict the target token t_t given the neighbor tokens $[t_i, .., t_{i+w}]$ in a certain window w. This approach is called the Skip Gram Model.

   In our project we used pre-trained vectors obtained from the Skip Gram model. It is important to note here that each token has exactly 1 **300-dimensional** word embedding, which is context independent. For example, the word embedding for the token *"bank"* would be the same in the sentence: *"The bank is closed today"* and in the sentence: *"I took a walk by the bank of the river"*.

2. ELMo: The ELMo embeddings that we use are the **1024-dimensional** learned vectors of the last layer of a deep shallow bidirectional language model (biLM), which is pre-trained on a large text corpus [maybe here be specific – don't know where exactly they were pretrained]. ELMo's model has two different LSTM layers; the first reads the input tokens from left to right and the second from right to left. The word embeddings obtained by ELMo are contextual, deep and character based. Given two different contexts C_1 and C_2 and a target token t_t, then the word embedding v_{t1} obtained from C_1 for t_t is different than the word embedding v_{t2} obtained from C_2 for t_t. For example, the word embedding for the token "bank" would be different in the above example. Also, ELMo's character based vector representations allow it to obtain embeddings for out-of-vocabulary tokens unseen in pre-training.

3. BERT: BERT's model is similar to ELMo, in the sense that it produces deep and contextual word embeddings, but with a few defferencecs. It is important to note that in contrast to ELMo, which obtains the embeddings from a shallow bidirectional LSTM, BERT gets the embeddings by a truly bidirectional Transformer model. In ELMo's architecture, neither of the two LSTMs takes both the previous and subsequent tokens into account at the same time. Also, BERT uses word-pieces (e.g. playing $\rightarrow$ play + ##ing) instead of words. This is effective in reducing the size of the vocabulary and increases the amount of data that is available for each word. The embeddings that we get are the **1024-dimensional** learned vectors of the last layer of this deep bidirectional Transformer model.

After we obtain the word embeddings for each token in the Context C of the input sentence and the fact candidate tokens t_{f_1} and t_{f_2}, we need to find a way to decide which of the fact candidates fits "better" in C. Assume that $C = [t_1, t_2, ..., t_k]$. Then the vectors of each token in the context would correspond to the matrix $M_C = [v_1, v_2, ..., v_k]$, where for each $1 \leq i \leq k$, v_i corresponds to a vector of dimension $d \in \{d_{glove}, d_{elmo}, d_{bert}\}$ obtained by model $m \in \{glove, elmo, bert\}$. Also the vectors of the candidate tokens t_{f_1} and t_{f_2} are denoted by v_{f_1} and v_{f_2}.

Define as $a_C = \frac{1}{k}\sum_{i=1}^{k} v_i$ the d dimensional vector obtained by taking the average of the vectors in M_C. Intuitively, this vector represents the average contextual meaning of the context C in the d dimensional embedding space. To model how well a candidate token"fits" in a context C, we measure the cosine angle between v_f and a_C. The cosine of the angle would measure how similar those two vectors are. A small angle signifies high similarity and a large angle low similarity. Define a similarity function

$$s(v_f, a_C) = \frac{v_f \cdot a_C}{\|v_f\|\|a_C\|}$$

which gives the cosine similarity between two vectors. The domain of s is $[-1, 1]$. Thus to turn the cosine similarity between two vectors into a valid probability function, we define the probability of two vectors being similar as

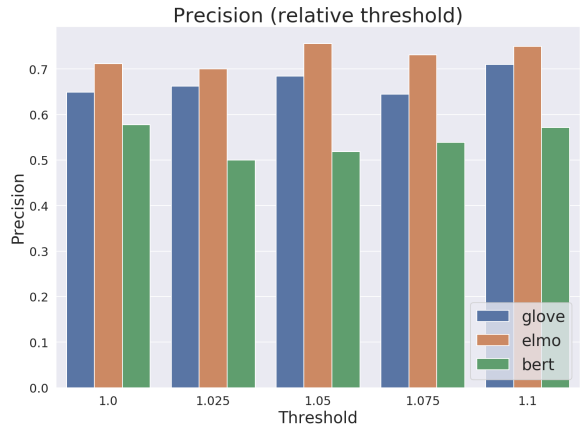
$$P(v_f, a_C) = \frac{s(v_f, a_C) + 1}{2}$$

Since we have a measure of the probability of a candidate token t_f with a vector v_f belonging in a Context C with vectors M_C, we can now find the best "suited" candidate vector among choices t_{f_1} and t_{f_2}. Intuitively, if $P(v_{f_1}, a_C) > P(v_{f_2}, a_C)$, then t_{f_1} "fits" better in the context C than t_{f_2}. We use the threshold models outlined above in order to increase the confidence in updates.

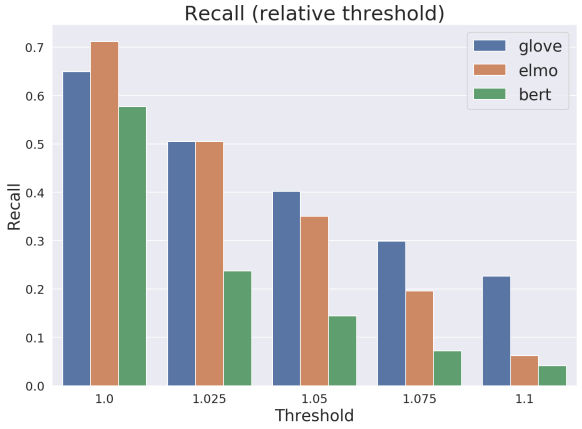
5. Evaluation

In order to evaluate the utility of language models on our fact-checking task we created a small structured (see section 8. Limitations) data-set of sentence pairs. The sentences come from random fact domains (sports, countries, languages, etc.) and among every pair one is correct. We measure precision and recall based on the incurred true positives and false positives (as outlined above). As unified metric we compute the $F_{0.5}$ score, a biased harmonic mean between precision and recall favoring precision. Results were recorded for Bert MLM as well as the three embedding models using Bert, GLoVe and ELMo over both the relative and absolute threshold model with varying threshold parameters.

Figure 1 shows the precision (a) and recall (b) of the embedding models using the relative threshold.



(a) Precision for Embedding models



(b) Recall for Embedding models

Figure 1. Embedding based models using relative Threshold

The absolute Threshold model shows similar trend in recall decrease, but lower recall and higher precision than the relative Threshold model. This is expected, as the absolute model imposes a minimum confidence in order to update which naturally reduces both true and false positives. However, precision increasing implies that more false than true positives were filtered, aligning with the intuition that false facts generally have low probability. ELMo's peak precision for the absolute model was reached at T=0.025 with 78% precision and 41%recall, whereas GloVe reached 85% precision with 11% recall at T=0.1.

Bert MLM performed competitively over the data-set, but was only applicable to a subset of examples (73%), due to lack of dictionary representations (see Limitations). With a relative threshold T=1 (equivalent to absolute T=0) the model had 71% precision and recall. For the relative TM precision and recall scaled from 72% to 68% and 45% to 36% respectively (for T=1.025 to T=1.1). Increasing Thresholds did not benefit the precision, indicating that the true positives were based off smaller confidence margins than the false positives. For the absolute TM precision and recall scaled from 68% to 73% and 32% to 22% respectively (for T=0.05 to T= 0.2).

5.1. Discussion

The goal of applying an update heuristic is to purify an information store. Thus, precision is the most important metric as it determines the truth percentage of the store in the lime. In expectation the truth value of the information store will converge to the precision. Recall measures how much positive change is permitted by the heuristic, effectively governing the speed of convergence.

Increasing Thresholds had marginal effects on precision, yet drastically decreased recall, suggesting that smaller Thresholds are more favorable. The decrease in recall implies that correct and false facts were often distinguished only by small likelihood margins. Relative Thresholds generally outperformed absolute Thresholds, however it should be noted that absolute thresholds usually demand higher margins. The larger the margin the more confidence we put in our update decision. While ELMo had the largest peaks (in both precision and recall) as well as the highest precision, its recall drops significantly faster than the measured recall for GloVe. This indicates that GloVe's true positives are based off larger margins and hence elicit more stability. Consequently, on a larger or different data set GloVe might generalize better.

While the MLM approach (Bert) had much better recall at higher Thresholds compared to the Embedding models it suffers from limited applicability. However, embedding based models ignore the order of the fact within the context and potentially lose predictive power by partially disregarding the sentence semantics. In contrast, the position of the fact matters for the Bert MLM approach as its embeddings

are deeply bidirectional. This to an extent explains the poor performance of the embedding based Bert model.

6. Dynamic Threshold Model

As our experiments results indicate, recall decreases linearly with increased Thresholds. While higher Thresholds supposedly reduce the number of False Positives they simultaneously reduce the amount of allowed True Positives, i.e. correct, beneficiary updates. Thus, the observed Precision of our models increased just very marginally, if at all with growing Thresholds. Consequently overall, the $F_{0.5}$ score decreases. This suggests that lower Thresholds are generally more efficient, as an update heuristic should also allow positive updates in order to be useful. A trivial, always safe heuristic would be to disallow any updates while making no progress.

An intuitive approach to compromise between avoiding False Positives, yet lessening the impact of high Thresholds is to employ dynamically scaling Threshold models. Ideally, for each update it were possible to derive an individual threshold. The absolute Threshold model already captures a minimal confidence in the update, whereas the relative Threshold model captures an increase of "truth value". However, as discussed in section 3.2 a relative Threshold model will judge two highly probably states more strictly, which may be counterproductive if these in fact correspond to "similar" results. Note, that this interpretation of similar is entirely qualitative and not absolute truth. For example, it may be argued that mistaking Munich for the German capital instead of Berlin is a less grievous mistake than choosing Athens as the capital. An approach to capture this relation is to scale Thresholds based on the similarity between two compared states. This effectively attempts to treat the truth value of facts as continuous space rather than through a definitive 0/1 loss function.

We capture this qualitative truth model for the embedding based language models. If two sentences are qualitatively similar we conjure that the embeddings that the language model produces are similar as well. Intuitively, qualitatively "similar" proposed facts should have similar embeddings as they appear in shared contexts and thus are likely to have joint features. In the high dimensional vector space that the embedding representation spans the similarity between two tokens can be interpreted as the angle between their respective embeddings, allowing us to use the cosine similarity between the latter as a similarity metric. Using this, we can scale the Threshold inversely proportional to the similarity. By assumption False Positives over "very similar" tokens incur less loss than False Positives over very different tokens. Thus, for tokens with low similarity we raise the Threshold as we require more certainty in order to confidently allow the risk of incurring a False Positive.

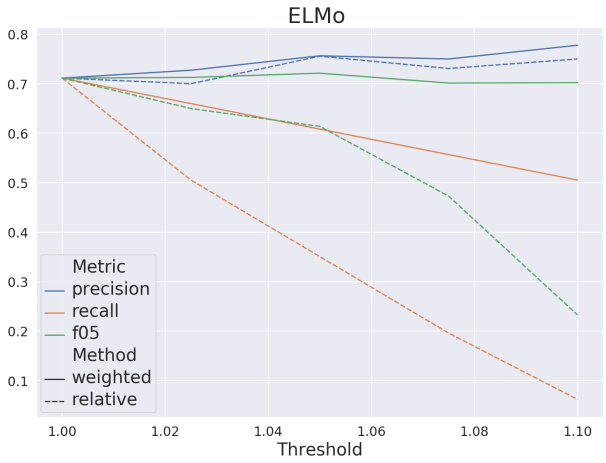
The dynamically scaled Threshold model is captured by modifying the Threshold T from previous models as follows (exemplary for the relative TM):

$$dynTM \triangleq \frac{P(F_2|C)}{P(F_1|C)} > w \cdot t + 1$$

where t is the Threshold offset of T and w scales inversly to the normalized cosine similarity between the Embeddings (denoted as E[]):

$$t = T - 1 \qquad w = 1 - \frac{cossim(E[F_1], E[F_2]) + 1}{2}$$

The dynamic approach was motivated to reduce the average Threshold in order to gain higher recall. The trade off for this is to incur potentially more False Positives with the intuition, that many of these False Positives are benign mistakes. Below we compare the performance of Elmo embeddings using the static relative Model (dashed lines) and the dynamic relative Model (solid line). For both cases we report results for varying Thresholds - in the case of the dynamic model this Threshold is the upper bound.



Using the dynamic Threshold model significantly increases recall while slightly improving precision compared to the static model. The $F_{0.5}$ score is now approximately constant, indicating that the dynamic model linearly trades off precision and recall (ratio 2:1) rather than suffering a collapse with increased Thresholds. While recall is expected to rise with dynamic thresholds as we reduce the Threshold on average, a rise in precision indicates that the scaling of thresholds did not allow more False Positives than previously. This is surprising given that the premise of the dynamic model was to potentially allow more False Positives but limit the holistic gravity of the mistakes. Consequently this suggests, that even for the absolute loss interpretation it is recommended to use the dynamic Threshold approach.

7. Limitations

While the obtained results are promising, there are several functional limitations that restrict the straight-forward application of this approach. For one, our sentence model assumes that facts are single token which is a problem for MLM models. In Bert MLM multiple masked tokens are not possible and masking just parts of a multi part token creates different contexts that affect the conditional probabilities (effectively introduces a bias). Another issue are limited dictionaries associated with the Language Models. Language models learn relationships over a fixed sized defined set of words of a language (English in our case), yet facts may take on arbitrary forms such as names. These tokens have no direct correspondence to an embedding but are broken up into syllables, resulting in a loss of meaning. Embedding based models are more flexible to handling this as they are often character based, but the questions remains whether the decisions are meaningful. Similarly to unknown tokens, rare tokens can affect the veracity of the updates. Rare tokens will have appeared in fewer contexts, resulting in lower embedding relations within the language model. Consequently, a rare fact may be deemed unlikely and discarded regardless of truth value.

A common problem in more general settings, unlike our constructed test set, is the sufficiency of context and/or the substance of the fact. For example, a sentence like "He was the best" both lacks definition of the fact as well as supplementary context. Updates to such sentences require a larger context, i.e. an enclosing paragraph. Paragraph based approaches can be starting points for extending fact-checking confidence, but are not immediately applicable to our sentence model.

A concern that we partially addressed in the qualitative truth setting is the generalization or specialization of facts. Updates that generalize, such as "Berlin is the capital of Brandenburg" to "Berlin is the capital of Germany" are not factually incorrect, yet may be allowed or disallowed at arbitration (likewise for the reverse specialization case).

Lastly, validating the accuracy of the language model based fact-checking approach is limited by the existence of suitable, labeled data, stemming both from the current lack of practicability as outlined above as well as the lack of such data sets. On a large unlabeled Wikipedia update stream data-set the Bert MLM model was applicable to only 25% of the data. With the (generous) assumption that most updates were from wrong to correct facts the MLM model had a 55% precision (not accurate as the assumption is incorrect). Yet, a large quantity of these decisions could have been arbitrary, as the sentences had non-practical forms (i.e. random names, rare tokens, lack of context and substance). Evaluating the performance of our explored models on a larger, dedicated data-set would be useful.

8. Conclusion

In this project we examined whether Language models can be used for Fact Checking by employing different existing pre-trained language models for an update stream based threshold model. We considered a Masked Language Model (BERT) based approach as well as a word embedding based approach comparing GloVe, ELMo and BERT embeddings. Our results show, that in expectation, the models can be used in order to purify the average truth in our dataset. We outlined the (dis-) advantages of different threshold models and confirmed these intuitions by measuring both precision and recall over a small dedicated data-set. The approaches we examined are surprisingly successful (with precision ranging from 68% to 85%) in identifying factual mistakes, considering the language models were not subject to any additional training. While we are carefully optimistic about the results, there remain several limitations to the applicability of the approach. We believe that a probabilistic framework based on Language Models could be improved further, in order to materialize into a concrete autonomous Fact Checking System. Some future work could address pre-training of the Language Models on a News Corpus (compared to our current Models which are pre-trained mostly on Wikipedia. Another issue to address is the long-tail problem which arises from the fact that Language Models learn the token distributions of the common topics included in the pre-training dataset rather than rare tokens. In order to address such an issue, one would either require an external Knowledge Base or a method capable of scaling training to even larger information sets such as the entire web.

References

Ciampaglia, G. L., Shiralkar, P., Rocha, L. M., Bollen, J., Menczer, F., and Flammini, A. Computational fact checking from knowledge networks. *PLOS ONE*, 10(6):1–13, 06 2015. doi: 10.1371/journal.pone.0128193. URL https://doi.org/10.1371/journal.pone.0128193.

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

Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jimenez, D., Gawsane, S., Hasan, S., Joseph, M., Kulkarni, A., Nayak, A. K., et al. Claimbuster: The first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10(12):1945–1948, 2017.

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., and Choi, Y. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2931–2937, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1317. URL https://www.aclweb.org/anthology/D17-1317.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vlachos, A. and Riedel, S. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pp. 18–22, 2014.