A Comparison of Instance Attribution Methods

Comparing Instance Attribution Methods to Baseline k-Nearest Neighbors Method

Evan de Kruif

Supervisor(s): Avishek Anand, Lijun Lyu, Lorenzo Corti

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 29, 2023

Name of the student: Evan de Kruif
Final project course: CSE3000 Research Project
Thesis committee: Avishek Anand, Lijun Lyu, Lorenzo Corti, Marco Loog

Abstract

In this research, a comparison between different Instance Attribution (IA) methods and k-Nearest Neighbors (kNN) via cosine similarity is conducted on a Natural Language Processing (NLP) machine learning model. The format in which the comparison is made is by way of a human survey and automated similarity comparisons of representative vectors. The goal of this is to judge and compare the effectiveness of each method's results in the context of a human's language understanding and ability to determine if a fact is true or not. Through this research, it was found that for results obtained on the same input, IA methods were preferred 32.5% more often than kNN. It is also shown that this preference is not linked to the similarity between the IA results and the kNN results. Through these findings, it can be seen that when understood through the lens of human comprehension, IA methods are much more effective at generating a set of influential training points from the model's training dataset.

1 Introduction

As more and more Artificial Intelligence and Machine Learning tools are being made, the need to explain them also rises[1]. Following this need, various methods have been produced to audit these algorithms' decision-making. One such method is Instance Attribution (IA).

1.1 Instance Attribution

Instance Attribution is a method of Explainable Artificial Intelligence (XAI) that, at its essence, will tell you how influential different datapoints from the machine's training dataset are on its decision-making. Unique because it can link as far back as the training data, Instance Attribution is a powerful yet fragile way to help explain a machine learning algorithm[2].

An example that is present in the current news cycle would be using IA methods to find the artworks most influential in the generation of AI art[3]. All AI artwork generators are trained using pieces from existing human artists. This means that a piece of generated art might be derivative of a certain artist's style and previous works. However, due to the black-box nature of these algorithms, they will not receive any credit. Conceptually, IA methods make it possible to trace a generated piece of AI art back to the training data and give credit to the most influential artworks for the said piece.

1.2 Research Question

In order to further this relatively new field of Instance Attribution, the research question that will be answered is: How do different instance attribution methods compare to the simplest instance attribution method, k-Nearest Neighbors (kNN), in the representative space? This question is worth answering because it is important to know what method to use when doing instance attribution so that you can identify what part of the training data is responsible for its decisions.

Using instance attribution, it can further be explained how a machine is making its decisions and whether it is arriving at the correct conclusion for the correct reason. As opposed to stumbling across the correct conclusion as a result of a misunderstanding derived from the training data.

It is essential to compare different IA methods and kNN because it is the most naive method of doing instance attribution. Using it as a baseline for comparison is reasonable, and can be used to show the relative improvement between other methods.

1.3 Existing Work

There are previous works on instance attribution, with methods such as FastIF[1][4] and TracIn[2][5] being created. Additionally, there are also works that are closer to a meta-analysis on the use of influence functions in Natural Language Processing (NLP), and NLP as a whole. From the broader use of NLP in identifying Fake News articles[1], to the specific uses of influence functions for shining a light on black box predictions[6] [7].

The opposing perspective also has had research done. Such as the drawbacks of using influence functions to improve deep learning algorithms[2]. This existing work includes one particularly relevant paper that compares different methods of instance attribution to one another in terms of algorithmic improvement and correctness[8].

What is missing from these papers is a comparison to the most naive method of instance attribution, k-Nearest Neighbors via cosine similarity, as well as a human perspective on how relevant the results from the influence functions are to the provided input. This is the gap that the research question is striving to fill.

1.4 Results Summary

As a result of this research's human survey, it can be seen through the data that influential datapoints are approximately 32% more preferable than when retrieved using IA methods as compared to kNN. Furthermore, the data shows that the similarity between IA results and kNN results, when measured by their representative vectors, does not influence whether they are preferred or not.

The takeaways are that instance attribution methods, at least FastIF and TracIn, are a more effective way of generating a set of influential datapoints for an NLP model when measured through human language comprehension. An

[1] Fast Influence Function

[2] Tracking Influence

additional takeaway is that instance attribution methods yield substantially different results than kNN. These differences in similarity, while improving the IA results compared to the kNN results, do not have a direct correlation with said improvement to preference.

For purposes of reproducibility and transparency, the implementation and brief setup instructions of the experiment environment are available here:
https://github.com/HatchIing/InstanceAttribution.

2 Related Work

Influence functions as a technique are not new. The vast majority of them are based on the statistic technique of up-weighting a training point a very small amount to see how model parameters change[9]. Kim et al (2015)[10] utilizes an approach that makes use of clustering to understand prototypes during training, though it doesn't study deep learning. The use of influence functions as an XAI method is fairly new, starting from Wojnowicz et al. (2016)[11] to the best of our knowledge.

Among the most referenced instance attribution methods are the ones described by Koh & Liang (2017)[6] and Yeh et al (2018)[12]. Koh & Liang[6] use the more traditional method from Hampel[9] to monitor the change in a prediction when a training point is dropped. Their method is also adopted by Guo et al (2021)[4] but is computationally expensive. This expense is because it involves the inversion of the Hessian matrix. Yeh et al[12] use the representer theorem which claims that the top layer of a network can be specified as a linear combination of values of the training points at the last layer.

Instance attribution isn't the only XAI method that has been researched though. Most closely related to instance attribution is feature attribution. Feature attribution shows the influence of the input features and how much weight each one has on the machine's decision-making[13]. The effectiveness of which has been measured by Nguyen et al (2021)[14], showing that it is also a valuable method for understanding and improving models.

As stated previously, instance attribution done by 'traditional' means, such as described by Koh & Lang (2017)[6], is expensive due to its scaling. As models become more advanced and accurate they also implement more parameters, which scale in $O(n^2)$[5] for time complexity. Combine this with the increasing scale of datasets which oftentimes encroach into the millions ($O(n)$[8] for increasing dataset size), it becomes readily apparent that newer, faster methods of instance attribution are needed if there is a hope of keeping up[4].

3 Problem Description

The research question of how different IA approaches compare against kNN in the representation space can be specified as comparative research. Through the course of this research, there isn't a new method being created, but rather existing ones are adapted to fit the experiment and analysis goes from there. This presents several problems that must be solved in order to adequately answer the question.

3.1 NLP Model

The first problem is how to adapt the pre-trained model that is used, ExPred[15], and the instance attribution methods that need to be compared. It is necessary to use a pre-trained NLP model in this research for convenience, consistency, and to remain within the time constraints. To this end, a pre-trained version of ExPred was provided and used.

ExPred is a two-part model based on the 'Explain and Predict, and then Predict Again'[15] paper. It is multifaceted, but for this application, only the fact-checking capabilities will be used. The first part of ExPred is an explanatory model that focuses on providing an explanation for the decisions the model makes, helping users understand how and why the model is making the choices it is making. The second part is the classification model that, in the end, will give the classification of whether the input claim and evidence is 'Supported' or 'Refuted'.

The two-part structure of the pre-trained model is useful for providing explanations, but it does make it more troublesome to follow the gradients, even if only oracle access is required. This is because the gradients of the classification need to be used and not necessarily the ones from the explanation.

3.2 IA Methods

The core of this research is the instance attribution methods, FastIF and TracIn. Both are convenient in one way because in order to get the necessary output of a list of training datapoints and their influences, all that is needed is the model itself, the input for the model, and the training dataset. In this case that would be the ExPred model, a datapoint formatted as an ExpredInput, and the FEVER[16] dataset.

Also convenient is that the metrics by which FastIF and TracIn measure influence do not matter. They only need to be pertinent when relative to themselves in order to return a 'most to least' influential subset of training datapoints.

FastIF

The first chosen instance attribution method is FastIF[4]. FastIF is a method that uses simple modifications to influence functions to significantly improve the run time. Apart from the efficiency optimizations, the actual influence function used by FastIF is very similar to the one described by Koh et al. (2017)[6]. The main optimization used by FastIF

is to narrow down the search space by using kNN. This would make the final result a subset of kNN, so the kNN optimization was cut from the pipeline. This makes the run time slower, but for the purposes of this experiment it is necessary.

TracIn

The second instance attribution method is TracIn[5], a method that tracks how loss on test points changes during the training process whenever the training example of interest is utilized. This method is also loosely based on the method described by Koh et al. (2017)[6], however instead of computing the inversion of the hessian matrix, which has a size that is quadratic in the number of model parameters, it uses random projections. This allows it to use sketches of the loss gradients which can be reused to compute randomized, unbiased estimators for the influence of given test points[5].

4 Methodology

The foundations for this research are the pre-trained ExPred model[15] and the FEVER[3] Dataset[16]. The different methods, k-Nearest Neighbors, FastIF, and TracIn, are able to be adapted to an NLP Fact-Checking context by using these as a base. This section will go over these foundations, methods, and how they work individually. The implementations and specifics on how they are adapted can be found in section 5.

4.1 FEVER Dataset

The FEVER dataset[16] is a large claim/evidence-based dataset with 98,000 annotated datapoints in the training set alone. Specifically, the Eraserbenchmark version[17] is utilized because it has been modified for usability by simplifying the datapoints and removing superfluous annotations that would otherwise be clutter in this context.

Each datapoint is labeled either 'SUPPORTED' or 'REFUTED', and contains annotated context for the claim in the dataset along with the required identifiers. The provided context is evidence relevant to the claim in the datapoint, usually pulled from the corresponding Wikipedia article about the same topic. Coupled with the fact that ExPred is trained using the FEVER dataset, it is perfect for the use case of this research.

4.2 k-Nearest Neighbor

The baseline instance attribution method is k-Nearest Neighbors through cosine similarity. K-Nearest Neighbors is a method of grouping similar datapoints based on their distance from the origin datapoint, then creating groups of size k.

The cosine similarity can be expressed as:

$$similarity(A,B) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 1: The formula for finding the cosine distance between datapoints A and B. This formula is used for origin datapoint A and on all other datapoints B. Sorting these distances and taking the closest k datapoints yields the k-Nearest Neighbors.

The cosine similarity is also used when comparing the representative vectors of the instance attribution and k-Nearest Neighbors results. The comparison between representative vectors is used to show that there is an appreciable difference between the two, and to see whether or not there is a correlation between their preference and similarity.

4.3 FastIF

As shown in Figure 2, the workflow of FastIF is a three-step process.

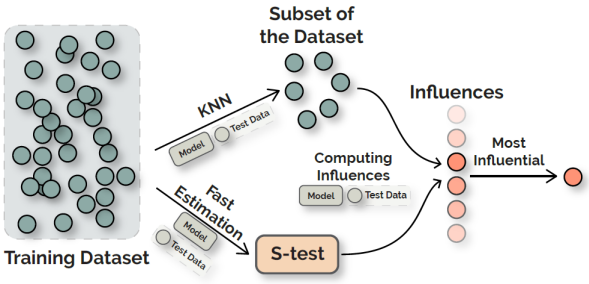

Figure 2: A high-level overview of the FastIF pipeline. Showing a subset of datapoints being selected from the training dataset via kNN. Then the inverse Hessian-vector product is estimated. Then the influence of the datapoints is calculated and the most influential points are given.

For the purposes of this research, the first step where a subset is chosen based on Facebook AI Similarity Search (FAISS)[18] kNN is omitted for two reasons. The efficiency of the influence function is not being analyzed, and in order to avoid having the result simply be a subset of the kNN via cosine similarity results.

The second step that FastIF takes is to estimate the inverse Hessian-vector product through the s-test. This can be represented generally through the equation in Figure 3.

$$s_{\text{test}} = H_{\hat{\theta}}^{-1} \nabla_\theta L(z_{\text{test}}, \hat{\theta})$$

Figure 3: This equation shows the value that FastIF pre-computes for each test datapoint z_{test}. H_θ^{-1} is the inverse hessian product that is calculated, $\nabla_\theta L$ is the calculated loss for z_{test} and input θ[4]

Finally, FastIF will use the pre-computed s-tests to calculate

[3]Fact Extraction and VERification

the influence of each datapoint as represented in Figure 4.

$$\mathcal{I}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_\theta L(z_i, \theta)$$

Figure 4: Equation for how FastIF calculates the influence of input z. s_{test} is the s-test described above. $\nabla_\theta L$ is the calculated loss between every iterated test point z_i and input θ.

At this point, FastIF would also be collecting the different influences from however many parallel processing threads it created and averaging them in order to output the final results. However, in this research, it was not possible to use parallel processing[4] so this step is effectively omitted due to only one thread being used. So there is no need for further computation.

4.4 TracIn

TracIn finds the influence of specific training examples by summing up the loss in all iterations that it was used. In order to approximate the idealized influence of a particular training example on a given test example, that is induced by the training process whenever the training example is utilized, TracIn uses the equation shown in Figure 5.

$$\texttt{TrackInIdeal}(z, z') = \sum_{t:\, z_t = z} \ell(w_t, z') - \ell(w_{t+1}, z').$$

Figure 5: The summation of loss gradients ℓ through all iterations of weights w_t and a fixed test point z'. The initial parameter vector before starting the training process is w_0, and the final parameter vector is w_T .

Simply approximating the idealized influence in this way would require TracIn to keep track of parameters, all training points used at each iteration, and essentially replay the training process. In order to avoid this, TracIn uses a checkpoint technique for practical heuristic influence, as described in Figure 6.

$$\texttt{TrackInCP}(z, z') = \sum_{i=1}^{k} \eta_i \nabla \ell(w_{t_i}, z) \cdot \nabla \ell(w_{t_i}, z')$$

Figure 6: Equation describing TracIn's checkpointing method for calculating approximate influence. Shows summation of the dot product between the loss gradient $\nabla \ell$ of test point z' and input point z, multiplied by n_i where n_i is the step size of iteration i.

Doing so allows the first-order approximation of the influence to approximate the parameter vector at the specific iteration where a given training example is visited. This does assume that the step size is kept constant between checkpoints and that each training example is visited only once.

[4]See Section 8: Discussion for full explanation.

4.5 Comparison Methodology

The general method that the research uses is to adapt the instance attribution methods so that they are able to take ExPred as the input model, and FEVER as the training dataset that is to be targeted. After the instance attribution methods are modified to these requirements, datapoints from the validation set of FEVER are chosen to be run. Along with the results from the k-Nearest Neighbors, the top influential training points are then taken from each.

Automated Comparison

The results from the instance attribution methods are compared to the results from kNN by the cosine similarity of their representative vectors. By doing this the semantic difference between the two can be seen. This means that it can be shown whether or not the instance attribution methods yield significantly different results from the nearest neighbors method.

This comparison is then combined with the results from the survey and analyzed. The goal is to observe whether there is a correlation between the similarity of the methods and how they fare in the human survey portion of the research.

Manual Comparison

For the manual comparison, the instance attribution and kNN results are trimmed down to the top 5 so that the subsets are more manageable to a human when reading them. Afterward, they are further formatted to where they only show the claims and the labels, along with the original input claim from the validation dataset.

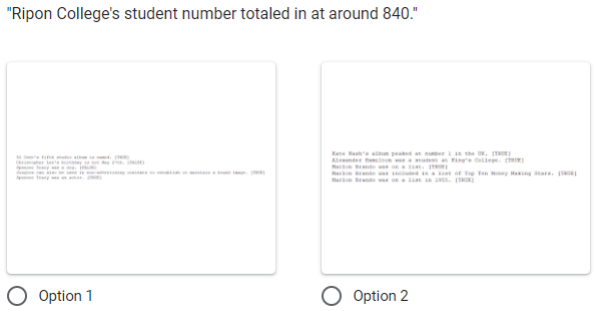

Figure 7: An example of a question from the survey. The provided claim from the validation dataset is shown on top and the choice is between either the instance attribution results (FastIF or TracIn) or the kNN results. Options are randomized as well as question order. For every provided claim there are two questions using it, one comparing FastIF to kNN and one for TracIn and kNN.

This is then presented to human participants, in the form of a Google Forms Survey[19]. The survey will record the answers which will show the number of times one method was chosen over another. This shows a preference for either the instance attribution or k-Nearest Neighbors results.

The requirements for participants are that they have at least a basic understanding of computer science and machine learning, as well as being fluent in English. The participants are selected for the survey through convenience sampling and perform the survey individually.

The results of the survey are then analyzed to show preference between the instance attribution methods and kNN. Background information such as the similarity between the results, based on their representative vectors, and relative similarity is also compared to the results of the survey. Through this analysis, conclusions can then be drawn about how the results answer the research question.

5 Experimental Setup

The setup for the experiment can be generalized into four parts: the adaptation of the instance attribution methods, running the instance attribution methods, creating the survey, and administering the survey.

5.1 IA Method Setup

The first step to setting up the experiment was to adapt both methods to be able to take ExPred and the FEVER dataset as inputs.

FastIF

For FastIF, it is fairly straightforward to adapt the method itself to the model. By separating the classification module from the explanation module within the ExPred PyTorch Model object and only calculating the gradient of the classification module, the two-part layout of ExPred is no longer an issue.

In order to allow FastIF to take in the FEVER Dataset, a custom PyTorch Dataloader had to be made with a custom collate function that converted the datapoint into an ExpredInput. Additionally, the collate function does the pre-processing on it so that the datapoint could be given to the ExPred model when FastIF called it.

Apart from that, the adaptation of FastIF is mainly matching data types between it and ExPred. An example would be when FastIF needs to know the output of ExPred, a few lines had to be added to the code to extract the vector that ExPred outputs and make it usable for FastIF. Small changes such as this make up the majority of this portion of the setup.

TracIn

TracIn is less straightforward to adapt because it is more specialized towards instance attribution on an image classification model based on resnet[20]. However, most functionalities inside TracIn that had to deal with resnet were mainly optimizations that would make the computation faster. Meaning they could be subverted or removed without affecting the method itself.

There is one substantial functionality within TracIn that had to be subverted though. TracIn creates an idealized version of the input classification, which is doable with pictures. By using resnet and the fact that there is a set number of pixels it is possible to create a best-fit example based on the model weights. However, this does not work with natural language processing, so it had to be removed. It is most likely possible to adapt TracIn to where it can create an idealized version of the input claim, but it was outside the scope of this research and its constraints.

In the end, TracIn is implemented much the same way as FastIF. Though the specific pipeline for computing influences is different, much of it is interchangeable. Such as the method for getting gradients and iterating through the training data. The same custom DataLoader from the FastIF implementation was also used for the TracIn implementation. This is because the DataLoader only needs to be able to iterate through the training data and format each point to where ExPred accepts it, there is no interaction between the DataLoader and the method per se.

5.2 k-Nearest Neighbors

The implementation for k-Nearest Neighbors is very simple. The method takes in the input claim, a number for k, and the training dataset. It then transforms the input claim into a vector and compares it to all datapoints in the training dataset which are also turned into vectors as it iterates.

By using the cosine distance between the claim and all datapoints in the training dataset, the closest points are able to be selected and returned as the results.

5.3 IA Methods Execution

The second step in setting up the experiment is to restrict the training dataset for each input selected from the validation dataset. This is done purely because of time constraints. To do this, a balanced subset of the original training dataset was created based on their direct relation to the input claim. Then a random sample of constant size from the training data is appended to it.

The related subset is created by searching the training data for keywords that appear in either the claim or the evidence. The search is done for specific keywords to avoid redundancy with kNN.

After all relevant datapoints are collected, the subset is balanced by removing either supported or refuted claims until they are equal. This is to avoid over-representing either side and accidentally influencing the instance attribution methods. Afterward, a random sample of 300 datapoints is taken from the training dataset. Combined, these two subsets create the trimmed set that is fed to the IA methods.

After the IA methods are run on the trimmed datasets, the automated comparison takes place using the same cosine similarity function from the kNN implementation. All sets of influential datapoints from both FastIF and TracIn are compared to the sets from kNN via cosine similarity and the results are stored for analysis.

5.4 Survey Creation

The third step is to compile the results and use them to create the user study. To do this the top five influential points were taken for each of the 10 claims. Since there were 10 claims that were run, between the three methods there are 30 sets of top 5 influential datapoints. These were then formatted by trimming everything but the claim and the label. Leaving each influential datapoint in the form of 'claim [label]'. These trimmed results were then used to create the survey in Google Forms.

In the survey, participants are presented with the claim, and two options in the form of a multiple-choice question as shown in Section 4. One option is either the formatted results of FastIF or TracIn, and the other option is kNN.

The participants are asked to: "Choose the set of sentences that would influence your decision making toward saying True or False." and were informed that "The given claim will not always be true.".

5.5 Administer Survey

Finally, the last step is to administer the survey to participants. Participants for the survey all had a baseline understanding of computer science and machine learning algorithms.

In total there were 20 participants that were selected via convenience sampling. The survey was set up to not record any data apart from the answers to the questions. Additionally, the survey was set to randomize the order of the 20 questions and to randomize the order of the two options in each question.

During the survey, participants were instructed to stay in contact in case they had any questions or if a survey question was unclear. Most participants were brought into a voice call as they either had questions about how to answer or because the compression that Google Forms uses on images made them difficult to read on their end. After all the surveys were administered, Google Forms automatically shows the statistics for the results.

5.6 Metrics

The metrics that the results will be analyzed on are each method's preferences from the survey and the cosine similarity.

Preference is shown through the survey from how often the respective option was picked. Every question is a choice between either FastIF/TracIn and the kNN results, so the frequency that one is picked over the other shows their preference.

The similarity is shown through cosine similarity between FastIF/TracIn results and kNN results. All results are converted into a representative vector and compared as described above.

6 Results

The results of the survey show that the difference between the instance attribution methods is relatively negligible. However, they are much preferred to the subset generated by kNN.

Comparing the cosine similarity of the FastIF and TracIn results with the kNN results shows that there is no relation to the similarity and whether either option was selected during the survey.

6.1 Survey Results

Table 1: Survey Results

QUESTION	FASTIF	KNN	TRACIN	KNN
BOWEN	0.7	0.3	0.8	0.2
CAMDEN	0.55	0.45	0.7	0.3
ILLINOIS	0.8	0.2	0.7	0.3
KESHA	0.65	0.35	0.8	0.2
PANDA	0.6	0.4	0.6	0.4
RIPON	0.55	0.45	0.5	0.5
SHADOW	0.65	0.35	0.75	0.25
SILVER	0.6	0.4	0.7	0.3
VIETNAM	0.75	0.25	0.55	0.45
WOZNIAK	0.6	0.4	0.7	0.3
	0.645	0.355	0.68	0.32

The questions are represented by an identifiable keyword from the given claim. Each pair of columns (FASTIF - KNN, TRACIN - KNN) show the percentage of times they were chosen for the respective question.

Table 1 illustrates the results of the survey. It is immediately apparent that the influence functions are preferred to kNN, with the influence functions being selected a combined 66.25% of the time.

Individually, TracIn was selected more than FastIF. However the difference is only 3.5% higher than FastIF, and because of the small sample size, it can not be reliably argued that one is preferred over the other. Therefore, stating that they are more or less comparable is a more conservative conclusion.

One thing that was seen through the results of the survey is that there was not a single question where kNN was preferred over the instance attribution option. At most, the preference closes in on being a 50/50 choice between the two. Meaning that the selection of one over the other is closer to being a random selection as opposed to one driven

by understandability.

This survey outcome shows that the best-case scenario for kNN is that its results are at the same level of preference as the instance attribution results. Though this only happens for the 'Ripon' question where it is close to 50% preference in both FastIF and TracIn comparisons.

An insight into why this could be the case is seen through the participants' responses when asked about their 'general thoughts' about the survey. There were some thoughts that were expressed by almost all of the participants:

- When neither option had something (keyword/direct relation to prompt) to do with the question. They preferred the option with a seemingly wider variety of information.
- When an option was more or less a rehashing of the same sentence (As is often the case with kNN results). They preferred the other option.

Both of these opinions lend themselves to choosing instance attribution the majority of the time.

6.2 Similarity Comparison

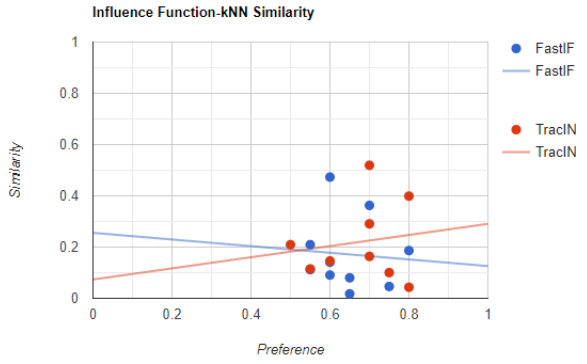

Figure 8: The y-axis shows cosine similarity between the influence function (Blue for FastIF. Red for TracIn) and the corresponding kNN results. The x-axis shows the preference of the influence function from the survey (I.E. .6 means it was picked 60% of the time over the kNN option). The blue and red lines are the best-fit lines for FastIF and TracIn respectively.

The similarity between the representative vectors of the influence functions and their corresponding kNN results is shown in Figure 8.

The scatter plot shows that in both cases the similarity and the preference do not have a strong relation, as seen by the best-fit line being almost horizontal in both cases. What can also be seen is that the overall similarity is low, with only one datapoint approaching a cosine similarity of 0.5.

Although the level of difference does not necessarily have an effect on the level of preference, it does mean that instance attribution methods not only yield different results than kNN, but also that those differences make them preferable to kNN.

7 Responsible Research

Due to the fact that this research required human participants, there are ethical considerations that need to be explained. Along with considerations about the participants' selection and privacy, extra consideration must be taken for what data if any is stored on them.

7.1 Human Participation and Considerations

The requirement for participants was that they had at least a baseline understanding of machine learning/computer science, and must be fluent in English. Finding participants that fit these criteria is not difficult as they are not overly specific, however, the research had to be done in quite a short amount of time. Due to the time constraints, it was necessary to reach out to people through personal connections, meaning that the participants were friends and family.

Though this did make it possible to finish the survey, even during the holiday season, it is something that needs to be stated. As in one way or another, the participants are not completely emotionally unbiased. This does mean that participants were solicited purely through personal connections and were not promised or given anything additional to participate.

The procedure for the survey can roughly be divided into three steps. Making contact with the participants, where their consent is gained and they are informed generally about the survey. The conduction of the survey itself, where participants fill out the survey. Then finally, a quick talk after the survey about their general thoughts and feelings about the survey and their answers.

Firstly, participants are asked if they would be able to fill out the survey. If yes, they are informed about what is expected from them. These expectations are outlined in the introduction of the survey as well. Additionally, they are informed about what information would be collected on them, which in this case is only their answers as that is the only thing pertinent to the research.

While filling out the survey, most participants were in a voice call as the compression that Google Forms uses made some of the multiple-choice options hard to read on their monitors. In the case an option was illegible they were sent the full-resolution picture through direct messages.

While in the voice calls, a concerted effort was made to not answer questions that by themselves or through their answers, would affect the way the participant answered the question in a biased way.

After the survey, almost all participants were asked about their opinions on it. It was also restated to most of them that nothing about their personal information is saved and only their responses are recorded, which themselves are also anonymous.

7.2 Replicability

If the research was to be reproduced there are several items that need to be considered: the selection of the participants, the restrictions on time, and the hardware requirements.

The selection of participants, as stated earlier, was done through convenience sampling in order to expedite the process and increase the chance that they accept to participate. This does lead to a personal connection with the participants and could in one way or another influence the results of the survey.

The restriction on time affected more than just the selection of participants. The results of the influence functions were also restricted because it was impossible to run them on the entire dataset in a timely fashion. Details on these dataset restrictions are detailed in section 5: Experimental Setup.

Lastly, the hardware that was used is important because the functions are specific to graphics cards with CUDA cores. This decision was made to speed up processing time, and for simplicity when writing the code. The fact that CUDA cores are required does mean that if someone tries to run the program on a device without a compatible graphics card, it will not work.

7.3 FAIR Research

The FAIR Guiding Principles for Scientific Data Management and Stewardship by Wilkinson et al. (2016)[21] are guidelines that are widely used to standardize scientific data management. Following the guidelines is good practice when working with research that deals with software and data.

Findable
Findable means that the research's software, implementation, and metadata, are easily findable by both humans and machines. To facilitate this, all word is being uploaded to the TU Delft Repository, and the implementation has been made public on GitHub.

Accessible
Accessible means that the research's software, implementation, and metadata, are easily accessible and retrievable. Through the same channels that the research and related material was made findable, it is also accessible through the same channels.

Interoperable
Interoperable means that the research software is able to be used with other data or through different means. This is achieved simply by providing the source code for the implementation along with instructions on how to set up the environment. After the initial setup, it is entirely possible to swap out the different parts of the experiment for other ones.

Reusable
Reusable means that the research is able to be used and modified through standard means. This research is made reusable through the implementation instructions in the GitHub repository. Though it has to be restated that a graphics card with CUDA cores is required to run the experiment as is.

8 Discussion

A direct comparison for this research could not be found as other research relevant to instance attribution does not have the human component. While a direct comparison might not be possible, it is possible to say that the results of this research are supported by the general consensus that influence functions are an improvement over kNN methods for instance attribution[4] [5] [8].

Several aspects of the research could have been done differently. Given more time and resources it would have been possible to run the instance attribution methods on the entire FEVER dataset. Not being able to do so makes it so that the results have an inherent flaw because the instance attribution methods were run on a fraction of the dataset that the model was trained on.

For full context, the FEVER training dataset has 97,955 datapoints. That number is trimmed down to 350 datapoints between the relevant subset and randomly selected datapoints combined. Even with this massive reduction in dataset size, all query runtimes added up to around 20 hours.

Additionally, the time constraint could have been alleviated to some extent if parallel processing was able to be used. This was impossible however due to a bug in PyTorch. The bug made it impossible to utilize a DataLoader with anything other than 0 workers. Which means only the main process can be utilized.

In the future, possibly with a different version of PyTorch or with different hardware, it would be possible to up the number of workers and speed up the execution time. The only difference that would need to be made is to change the number of workers when declaring the DataLoader.

This research could also be improved in the future by selecting participants in a different manner other than convenience sampling. The use of convenience sampling was, again, due to time constraints. However, given more time it would be possible to find a bigger sample size of truly non-biased participants.

Most improvements to the research have to do with expanding the scope. Comparing more instance attribution methods, different nearest neighbor methods, using different

parameters, etc. Expanding the scope of the comparisons would allow for a broader, but also more detailed, analysis.

9 Conclusions and Future Work

The results show that instance attribution methods, specifically FastIF and TracIn, are better methods at generating an understandable set of influential training datapoints than k-Nearest Neighbors via cosine similarity. The 32.5% difference in favorability between the two methods illustrates that there is a significant improvement. Even though instance attribution methods are more expensive at run-time, they yield a better result.

It can also be concluded that the similarity between the influential training datapoints from the instance attribution and the k-Nearest Neighbors methods did not have an effect on their favorability during the survey. This means that regardless of the similarity, the k-Nearest Neighbors results are not preferred over instance attribution. At most, they will exhibit similar favorability.

Future work for this research would be to compare the increase in preference to the increase in time complexity for different methods. A major restraining factor during this research was the time investment in running the instance attribution methods. So the logical question to ask is at what complexity, dataset size, model depth, e.t.c., does the increase in usability given by instance attribution get out weighted by the increased resource and time investments.

References

[1] T. Traylor, J. Straub, Gurmeet, and N. Snell, "Classifying fake news articles using natural language processing to identify in-article attribution as a supervised learning estimator," *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pp. 445–449, 2019.

[2] S. Basu, P. Pope, and S. Feizi, "Influence Functions in Deep Learning Are Fragile," *arXiv e-prints*, p. arXiv:2006.14651, June 2020.

[3] Z. Epstein, S. Levine, D. G. Rand, and I. Rahwan, "Who gets credit for ai-generated art?," *iScience*, vol. 23, no. 9, p. 101515, 2020.

[4] H. Guo, N. F. Rajani, P. Hase, M. Bansal, and C. Xiong, "Fastif: Scalable influence functions for efficient model interpretation and debugging," 2020.

[5] G. Pruthi, F. Liu, M. Sundararajan, and S. Kale, "Estimating training data influence by tracing gradient descent," 2020.

[6] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," 2017.

[7] X. Han, B. C. Wallace, and Y. Tsvetkov, "Explaining black box predictions and unveiling data artifacts through influence functions," 2020.

[8] P. Pezeshkpour, S. Jain, B. Wallace, and S. Singh, "An empirical comparison of instance attribution methods for NLP," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Online), pp. 967–975, Association for Computational Linguistics, June 2021.

[9] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 383–393, 1974.

[10] B. Kim, C. Rudin, and J. Shah, "The bayesian case model: A generative approach for case-based reasoning and prototype classification," 2015.

[11] M. Wojnowicz, B. Cruz, X. Zhao, B. Wallace, M. Wolff, J. Luan, and C. Crable, ""influence sketching": Finding influential samples in large-scale regressions," 11 2016.

[12] C.-K. Yeh, J. S. Kim, I. E. H. Yen, and P. Ravikumar, "Representer point selection for explaining deep neural networks," 2018.

[13] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 3319–3328, JMLR.org, 2017.

[14] G. Nguyen, D. Kim, and A. Nguyen, "The effectiveness of feature attribution methods and its correlation with automatic evaluation scores," 2021.

[15] Z. Zhang, K. Rudra, and A. Anand, "Explain and predict, and then predict again," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, (New York, NY, USA), p. 418–426, Association for Computing Machinery, 2021.

[16] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a large-scale dataset for fact extraction and VERification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 809–819, Association for Computational Linguistics, June 2018.

[17] J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace, "Eraser: A benchmark to evaluate rationalized nlp models," 2019.

[18] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[19] E. de Kruif, "Preference questionnaire," 2023.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[21] M. Wilkinson, M. Dumontier, I. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. Boiten, L. Da Silva Santos, P. Bourne, J. Bouwman, A. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. Evelo, R. Finkers, A. Gonzalez-Beltran, A. Gray,

P. Groth, C. Goble, J. Grethe, J. Heringa, P. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. Lusher, M. Martone, A. Mons, A. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons, "The fair guiding principles for scientific data management and stewardship: Comment," *Scientific Data*, vol. 3, 2016.