ReVera Framework: Um Framwork para rastreabilidade em fact-checking automático

João Victor de Souza[1], **Elias Cyrino de Assis**[1],
Fabrício Martins Mendonça[2], **Jairo Francisco de Souza**[2]

[1]LApIC Research Group – Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
36036-900 – Juiz de Fora – MG – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora – Juiz de Fora/MG – Brasil

{joao.souza, elias.cyrino, fabricio.mendonca, jairo.souza}@ice.ufjf.br

Abstract. *Professional fact-checking tends to be costly and scalability challenging. For this reason, a series of methods to automate this process has been emerging. However, these methods have been created with monolithic architectures, not making use of pre-built parts, and are harder to be understood by others. This work aims to propose a framework, where the development of methods can be done in a modular manner, creating workflows based on key steps that can be linked together to generate the input classification. The framework makes use of a proposed data traceability ontology to map all generated data during execution under a unified vocabulary, which facilitates communication between these independent components.*

Resumo. *A verificação manual de fatos tende a ser cara e a escalabilidade desafiadora, incentivando a busca por métodos automáticos. Porém, esses métodos foram criados com arquiteturas monolíticas, não fazendo uso de peças pré-construídas e são mais difíceis de serem compreendidos por terceiros. Este trabalho tem como objetivo propor um framework, onde o desenvolvimento de métodos pode ser feito de forma modular, criando fluxos de trabalho baseados em etapas-chave que podem ser interligadas para gerar a classificação de entrada. O framework faz uso de uma ontologia de rastreabilidade proposta para mapear todos os dados gerados durante a execução sob um vocabulário unificado, o que facilita a comunicação entre esses componentes independentes.*

1. Introdução

O consumo de informações tem sido significativamente alterado pelo notável crescimento das redes sociais, que vem se tornado bastante comum no cotidiano das pessoas [Shu et al. 2017]. A sua utilização como fonte de notícias e informação tem sido cada vez mais comum [Conroy et al. 2015]. Entretanto, nem todo conteúdo publicado é verídico [Souza et al. 2020].

No jornalismo, a verificação de fatos (do inglês, *fact-checking*), pode ser definida como uma tarefa que visa determinar a veracidade de uma informação com base em fontes externas confiáveis. Entretanto, este tem sido um problema cada vez maior devido à

quantidade de informações que os usuários precisam lidar [Shu et al. 2017], e isso acaba aumentando não apenas a demanda por verificação, mas também torna o processo cada vez mais custoso de ser realizado [Hassan et al. 2015]. Além disso, a velocidade com que as informações trafegam em redes sociais ou serviços de mensagens cria um desafio cada vez maior [Hassan et al. 2015].

A diferença entre o momento que uma informação é vista e compartilhada pelos usuários e as primeiras verificações pode ser longa demais para evitar os impactos negativos dessa disseminação. Esse efeito tem levado a uma busca por novas formas que possam diminuir cada vez mais o tempo necessário para esse processo de verificação [Shu et al. 2017]. Diversos métodos automáticos tem sido apresentados para tentar solucionar o problema. Porém, sistemas desse tipo tendem a ter um desenvolvimento complexo, devido a quantidade de fases e dados necessários para realizar a verificação [da Silva et al. 2020], fornecendo um resultado transparente ao usuário [Kotonya and Toni 2020].

Assim sendo, este trabalho visa fornecer um *framework* baseado em uma ontologia para a criação de métodos para *fact-checking end-to-end*, que são as abordagens que tratam todos os aspectos relacionados à verificação, com o objetivo de fornecer interoperabilidade entre componentes e dados de proveniência sobre o *pipeline* de processamento. As principais contribuições desse trabalho são: (1) mostrar uma ontologia para o processo de *fact-checking* automático que auxilia na reprodutibilidade e rastreabilidade; (3) apresentar um framework para a implementação de métodos de fact-checking automático com base em componentes reutilizáveis.

2. Trabalhos Relacionados

Na literatura, vários trabalhos lidam diretamente com o problema da verificação de fatos [Santos and Pardo 2020, Miranda et al. 2019, Gerber et al. 2015]. No entanto, de acordo com [Graves 2018], é importante automatizar não apenas a verificação, mas também etapas de identificação e geração de resultados. Além de verificar as declarações, também é importante monitorar fontes que possam gerar fatos que valham a pena validar, além de oferecer meios para que seu resultado seja divulgado e alcance as pessoas.

Nessa área, sistemas *end-to-end* são aqueles que implementam todos esses componentes do processo, tratando não apenas a coleta de evidências e avaliação dos dados, mas também aplicando formas de divulgação de resultados e monitoramento de fontes de mídia em geral (debates, entrevistas, redes sociais). Para combater a desinformação, [Hassan et al. 2017] e [Nadeem et al. 2019] usam esse tipo de sistema para construir suas abordagens. ClaimBuster [Hassan et al. 2017] destaca-se como um serviço que funciona de forma totalmente automatizada, monitorando e identificando as declarações que podem ser validadas, e informando o veredicto através de um portal na Web[1] e no Twitter. Por outro lado, FAKTA [Nadeem et al. 2019] não possui um monitoramento ativo de afirmações, mas também é capaz de realizar o processo automaticamente. Para isso, os autores utilizam-se métodos de aprendizagem de máquina para identificar o posicionamento relevante do documento em relação ao que se pretende verificar.

Mesmo que esses trabalhos consigam lidar com o problema de *fact-checking*,

[1] https://idir.uta.edu/claimbuster/

eles não são capazes de prover uma arquitetura aberta e capaz de interoperar com outros métodos já existentes, ou reutilizá-los. A utilização de componentes isolados e interoperáveis ajuda a evitar desperdícios com retrabalhos e acelerar o tempo de desenvolvimento [Both et al. 2016]. Na área de Processamento de Linguagem Natural, *Natural Language Processing - NLP*, foram encontrados dois usos pra esse tipo de técnica. Em [Volodina et al. 2012], essa técnica foi aplicada para o aprendizado de idiomas, onde foi criada uma arquitetura com base em *web services* onde os serviços trocam anotações entre si. Similarmente, [Both et al. 2016] também realiza uma tarefa semelhante para prover interoperabilidade em sistemas de *Question Answering*. Esse trabalho também se apoia em uma ontologia, a QA Ontology, que é utilizada para armazenar todas as informações geradas no processo de construção da resposta.

Ainda que esses trabalhos apresentem avanços em suas respectivas áreas, eles podem não se adaptar completamente à área de *fact-checking*. Durante o processo de verificação, é preciso coletar e e processar evidências para se estabelecer um veredito. Ao organizar semanticamente essas informações, através de uma ontologia que aborda todo o processamento, faz com que seja possível capturar melhor o conhecimento disponível [Munir and Sheraz Anjum 2018]. Na literatura, os trabalhos disponíveis descrevendo ontologia ou formas de anotação de dados para *fact-checking* não contemplam esse aspecto. Trabalhos como [Rehm et al. 2018] focam em criar uma forma de descrever os resultados obtidos pela verificação, e não detalhar o processo de verificação em si. O presente trabalho, por sua vez, apresenta uma proposta de framework para abordagens de *fact-checking* que permite auxiliar na reprodutibilidade de experimentos e na construção de novas abordagens através do reuso de componentes entre abordagens distintas. Uma vez que o processo de *fact-checking* é dividido em etapas que podem ser definidas e alteradas facilmente no framework, este permite também auxiliar no processo de experimentação com diferentes componentes. Ainda, para garantir a rastreabilidade de informações do processo de verificação, é apresentada uma ontologia que descreve as relações e entidades que fazem parte do processo de validação, e que pode usada por qualquer abordagem ou sistema fora deste framework.

3. Ontologia Revera

Para fornecer interoperabilidade entre os componentes da mesma etapa, foi desenvolvida a ontologia ReVera[2] para descrever todos os dados que podem ser gerados em cada uma das etapas do processo de verificação automática de fatos à partir de dados textuais. Assim, independentemente da implementação, é possível obter uma mesma organização semântica dos dados gerados. Através desse tipo de especificação, é possível intercambiar informações entre diferentes sistemas, através da utilização de um vocabulário em comum [Bittner et al. 2006].

A PROV Ontology (PROV-O) [Belhajjame et al. 2012] foi definida para ser simples e extensível pelas aplicações que irão utilizá-la, onde a partir dessa simplificação, já é possível criar representações específicas para um determinado domínio. Sendo assim, a partir dessas classes e relações, foi definida uma ontologia de proveniência, com base na PROV-O, para definir os dados de proveniência que podem ser gerados e consumidos durante o processo de verificação das etapas, definido no *pipeline* proposto.

[2]Origina-se do latim, onde *revera* significa “na verdade”

A ontologia desenvolvida, chamada de ReVera Fact Checking Ontology, referida através do prefixo *fc:*, mapeia todas as entidades que são geradas durante o processamento. Ela foi pensada para ser uma ontologia de proveniência de dados, onde os dados gerados em um determinado estágio da *pipeline* de *fact-checking* são propriamente identificados. Isso faz com que exista um certo nível de reprodutibilidade dos resultados obtidos, podendo ser identificados e reprocessados com quaisquer outras configurações de componentes. O diagrama da ontologia pode ser observado na Figura 1. Na representação, os círculos brancos representam as classes definidas na ontologia e os azuis indicam as classes herdadas da PROV-O. As arestas indicam as relações, onde a origem é o sujeito e o alvo é o objeto, sendo as arestas azuis relações herdadas, as vermelhas, definidas pela ReVera Ontology, e as pretas não nomeadas na figura indicam relações do tipo *is a* (é um/é uma).

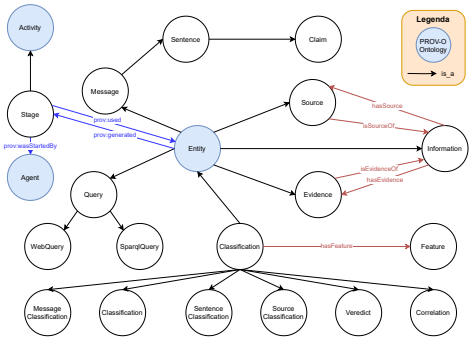

Figure 1. Representação de classes e relações da ontologia proposta

A ontologia proposta estende a PROV-O, especializando classes como Entidades e Atividades especificamente para representar o *pipeline* de processamento das afirmações.É importante destacar que, em uma abordagem de pesquisa recente, a FC Ontology foi desenvolvida no editor de ontologias Onto4ALLEditor[3], que possibilita a construção colaborativa de ontologias por mais de um usuário através da web e faz uma validação dos componentes ontológicos (classes, relações, propriedades e axiomas) construídos [Mendonça et al. 2020]. Essa característica atribuída à construção da ReVera Ontology contribuiu com a qualidade de seu conteúdo ontológico.

4. ReVera Framework

A fim de fornecer uma plataforma para o desenvolvimento de abordagens para a verificação de fatos, o *framework* foi desenvolvido com base nas etapas necessárias para serem implementadas em sistemas *end-to-end*, o ReVera Framework. Além disso, ele foi desenvolvido de forma a permitir o reuso, ser extensível e paralelizável.

Essa arquitetura necessita de um módulo de gerenciamento central para receber e gerenciar todas as solicitações de processamento. Cada uma das implementações das etapas possíveis, chamados de componentes, precisam se comunicar com o *core*, já que é necessária a entrada e saída de dados entre eles. Essa comunicação, entre os componentes e o core são realizadas através de um mensageiro. A Figura 2 mostra a arquitetura proposta.

[3]http://onto4alleditor.com/

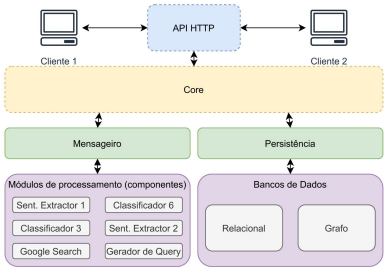

Figure 2. Arquitetura proposta para o *ReVera Framework*

O *core* é o módulo que precisa ter visão de todo o sistema. Ele irá receber todas as solicitações de processamentos, além de fornecer uma interface para a comunicação externa para se obter os resultados gerados pelo processamento. Para manter todas as informações, ele se comunica diretamente com dois bancos de dados específicos: um relacional e outro orientado a grafos. A base de dados relacional mantém o registro de todas as requisições de processamento, além de rastrear todos os componentes necessários para o processamento da requisição, identificando quais já foram concluídos e quais ainda precisam serem executados. Os dados gerados pela execução de cada etapa, descritos pela ontologia, são organizados em um grafo e salvos em um banco de dados na forma de triplas RDF (*Resource Description Framework*). Esse módulo é necessário para o sistema evitar que os componentes que realizam o processamento tenham que lidar com o encaminhamento e o controle de estado de cada uma das requisições.

Os componentes precisam ser especializados para realizar a tarefa necessária para que a etapa seja concluída. Ao ligar, em cadeia, componentes de diferentes etapas é possível transformar as sentenças ou triplas de entrada, encontrando os dados que o classificador final necessita para avaliar sua veracidade. Essas peças conectam-se ao núcleo através de mensageiros, onde para cada um dos componentes é criado uma fila de mensagens. Caso haja mais de uma instância de um mesmo componente, esses componentes escutam a mesma fila e isso permite paralelizar as mensagens que são inseridas na fila. A utilização desse tipo de arquitetura, muito utilizada em conjunto com microsserviços, permite um desacoplamento grande entre todos os módulos do *framework*, pois os módulos não precisam tratar da comunicação diretamente, não havendo restrições para a aplicação de diferentes plataformas ou linguagens de programação.

Ao ser implementada dessa forma, o *framework* apoia-se na ontologia ReVera para mediar o funcionamento de todos os componentes. Essa característica é o que cria uma padronização semântica entre todas as entradas e os resultados dos processamentos realizados. Isso acaba facilitando características como reuso, através da reutilização dos componentes, além de reprodutibilidade e rastreabilidade, que permitem que os resultados possam ser compreendidos e reproduzidos através das informações geradas durante o processo.

Nesse contexto, existe um desacoplamento entre os componentes, já que toda execução fica restrita a coerência entre os tipos de entrada e saída esperados para cada um dos componentes. Isso permite que a substituição de um componente por outro do mesmo tipo seja menos custosa, e que a sua reutilização para criação de novos *pipelines* seja mais simples, já que basta apenas considerar suas entradas e saídas.

Todos esses artefatos gerados durante o processamento são persistidos no banco de dados, permitindo que todo o fluxo de troca de informações possa ser analisado posteriormente. Uma das implicações diretas é a possibilidade de tornar o processo de reprodutibilidade mais transparente, podendo explicitar os dados que são consumidos e gerados. Desse modo, é possível consumir os dados de entrada para verificar se é possível obter tais dados, comparando diretamente com os resultados que foram obtidos.

A segunda implicação é permitir que os dados gerados possam ser rastreados, podendo analisar a transformação dos dados entre as etapas. Consequentemente, é possível obter todas as evidências e artefatos utilizadas para validar uma informação, desde a requisição até a finalização do processamento. Portanto, métodos que trabalham com a geração da explicação de forma automática podem se beneficiar disso como uma forma de obtenção dos dados.

Além disso, ao utilizar a estrutura já fornecida, são disponibilizados métodos para o gerenciamento e armazenamento dos fluxos de execução, utilizando um vocabulário pré-definido, podendo simplificar o uso dos dados de saída para diferentes sistemas.

5. Validação

Visando servir como uma prova de conceito para o *framework*, foi desenvolvido um *pipeline* baseado no mesmo trabalho anterior [Gerber et al. 2015]. Todas as etapas de processamento foram dividas em componentes de responsabilidade única dentro do *framework*, onde cada componente está responsável por uma forma de implementação da etapa de processamento. Além dos componentes que puderam ser identificados a partir de [Gerber et al. 2015], foram adicionados outros a fim de alterar a forma de processamento para ter como entrada documentos em texto de linguagem natural, ao invés de triplas RDF. Para isso, foi necessário alterar as etapas iniciais de tratamento da entrada.

Assim que a resposta é produzida, é responsabilidade de cada componente informar ao *core* o documento de resposta. Essa resposta é feita através do mensageiro, por uma fila específica, contendo o identificador do grafo, necessário para agrupar todos os dados relacionados a requisição e o próprio documento resultante. Assim, ao receber esses dados, o *core* é capaz de persistir os dados e prosseguir com o processo, identificando a próxima etapa e enviando todos os dados persistidos no grafo da requisição como entrada.

Analogamente, as outras etapas possuem protocolos de entrada e saída similares: todas procuram por instâncias de certas classes no documento de entrada e, ao finalizarem o processamento, gerar uma resposta seguindo o vocabulário da ReVera Ontology. Com isso, houve uma reutilização dos componentes já implementados e, através de uma reorganização do fluxo de execução, foi possível experimentar uma abordagem diferente.

Dessa forma, utilizando o *framework*, os componentes não precisaram lidar com os possíveis fluxos de dados que possam ocorrer, limitando-se apenas a compreender o vocabulário de suas entradas e saídas. Isso facilitou a extensão do trabalho de [Gerber et al. 2015] de maneira com que foi possível reaproveitar grande parte dos códigos já implementados.

6. Conclusões e Trabalhos Futuros

Este trabalho apresenta um *framework* e uma ontologia para rastreabilidade, a ReVera Ontology, para o desenvolvimento de métodos de verificação automática de fatos (*fact-checking*). O *framework* consiste de um *core* e módulos que implementam determinadas partes do processamento (componentes), é capaz de receber solicitações de processamento de diferentes *pipelines* e montar um grafo com informações de proveniência, seguindo o vocabulário da ontologia.

Além disso, o desenvolvimento do *framework* proposto nesta pesquisa pode ser considerado uma solução inédita para a área de *fact-checking*. Foram identificados na literatura trabalhos para *Question Answering* e *Language Learning*, porém não foram encontradas soluções desse tipo para *fact-checking*. A implementação forneceu uma forma desacoplada e distribuída para implementar sistemas de verificação, permitindo uma maior facilidade para alterar individualmente as partes do processo. Além disso, através do *framework*, foi possível entender o funcionamento de diversas pesquisas da área, gerando também uma classificação para as principais etapas realizadas durante o processamento.

Como trabalhos futuros, pretende-se implementar mais componentes para o *framework*, buscando novos trabalhos disponíveis na literatura e aplicando-os para que possam utilizar a ontologia. Novos componentes permitem a criação de novos *pipelines*, o que permite a evolução, verificando deficiências para mais casos de testes.

Além disso, devido a possibilidade de serem testadas diferentes combinações de componentes, futuramente podem ser adicionadas ferramentas de avaliação para os *pipelines* desenvolvidos sobre o *framework*. Elas devem permitir executar diferentes componentes e verificar, através de métricas de performance disponíveis na literatura, o desempenho de diferentes fluxos em um conjunto de testes, auxiliando em metodologias de experimentação dos resultados.

References

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. (2012). Prov-o: The prov ontology. Technical report.

Bittner, T., Donnelly, M., and Winter, S. (2006). Ontology and semantic interoperability.

Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., and Lange, C. (2016). Qanary — a methodology for vocabulary-driven open question answering systems. In *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678*, page 625–641, Berlin, Heidelberg. Springer-Verlag.

Conroy, N. J., Rubin, V. L., and Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

da Silva, F. R. M., Freire, P. M. S., de Souza, M. P., de A. B. Plenamente, G., and Goldschmidt, R. R. (2020). Fakenewssetgen: A process to build datasets that support comparison among fake news detection methods. In *Proceedings of the Brazilian Symposium on Multimedia and the Web*, WebMedia '20, page 241–248, New York, NY, USA. Association for Computing Machinery.

Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngonga Ngomo, A.-C., and Speck, R. (2015). Defacto - temporal and multilingual deep fact validation. *Web Semantics: Science, Services and Agents on the World Wide Web*.

Graves, D. (2018). Understanding the promise and limits of automated fact-checking.

Hassan, N., Adair, B., Hamilton, J., Li, C., Tremayne, M., Yang, J., and Yu, C. (2015). The quest to automate fact-checking. *Proceedings of the 2015 Computation + Journalism Symposium*.

Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jimenez, D., Gawsane, S., Hasan, S., Joseph, M., Kulkarni, A., Nayak, A. K., Sable, V., Li, C., and Tremayne, M. (2017). Claimbuster: The first-ever end-to-end fact-checking system. *Proc. VLDB Endow.*, 10(12):1945–1948.

Kotonya, N. and Toni, F. (2020). Explainable automated fact-checking for public health claims.

Mendonça, F. M., de Castro, L. P., de Souza, J. F., Almeida, M. B., and Felipe, E. R. (2020). Onto4alleditor: um editor web gráfico de ontologias direcionado a diferentes tipos de desenvolvedores de ontologias. *Proceedings of the XIII Seminar on Ontology Research in Brazil and IV Doctoral and Masters Consortium on Ontologies (ONTOBRAS 2020)*.

Miranda, S., Nogueira, D., Mendes, A., Vlachos, A., Secker, A., Garrett, R., Mitchell, J., and Marinho, Z. (2019). Automated fact checking in the news room. *CoRR*, abs/1904.02037.

Munir, K. and Sheraz Anjum, M. (2018). The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, 14(2):116–126.

Nadeem, M., Fang, W., Xu, B., Mohtarami, M., and Glass, J. (2019). Fakta: An automatic end-to-end fact checking system.

Rehm, G., Schneider, J. M., and Bourgonje, P. (2018). Automatic and manual web annotations in an infrastructure to handle fake news and other online media phenomena. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Santos, R. and Pardo, T. (2020). *Fact-Checking for Portuguese: Knowledge Graph and Google Search-Based Methods*, pages 195–205.

Shu, K., Sliva, A., Wang, S., Tang, J., and Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

Souza, J., Gomes Jr, J., Marques, F., Julio, A., and Souza, J. (2020). A systematic mapping on automatic classification of fake news in social media. *Social Network Analysis and Mining*, 10.

Volodina, E., Borin, L., Loftsson, H., Arnbjörnsdóttir, B., and Leifsson, G. Ö. (2012). Waste not, want not: Towards a system architecture for icall based on nlp component re-use. volume 80, pages 47–58.