The Workshops of the Tenth International AAAI Conference on Web and Social Media
Wiki: Technical Report WS-16-17

Wikipedia Knowledge Graph with DeepDive

Thomas Palomares
tpalo@stanford.edu

Youssef Ahres
yahres@stanford.edu

Juhana Kangaspunta
juhana@stanford.edu

Christopher Ré
chrismre@cs.stanford.edu

Abstract

Despite the tremendous amount of information on Wikipedia, only a very small amount is structured. Most of the information is embedded in unstructured text and extracting it is a non trivial challenge. In this paper, we propose a full pipeline built on top of DeepDive to successfully extract meaningful relations from the Wikipedia text corpus. We evaluated the system by extracting company-founders and family relations from the text. As a result, we extracted more than 140,000 distinct relations with an average precision above 90%.

Introduction

With the perpetual growth of web usage, the amount of unstructured data grows exponentially. Extracting facts and assertions to store them in a structured manner is called knowledge-base construction (KBC). It has recently received tremendous interest from academia (Weikum and Theobald 2010) with projects such as CMU's NELL (Carlson et al. 2010; Lao, Mitchell, and Cohen 2011), MPI's YAGO (Kasneci et al. 2009), Stanford's DeepDive (Zhang 2015), and from industry with Microsoft's EntityCube (Zhu et al. 2009) and IBM's Watson DeepQA (Ferrucci et al. 2010). Thanks to the recent development of these KBC systems, large databases such as DBPedia (Bizer et al. 2009) and YAGO (Kasneci et al. 2009) are able to automatically collect and store factual information about the world.

In an effort to support Wikipedia, the Wikimedia Foundation launched a similar knowledge base called Wikidata (Vrandečić 2012). Primarily based on crowd-sourcing, Wikidata stores structured information and makes them available to the world. Automatically populating it using a KBC system will help scaling it up and increase its impact around the world. In this paper, we describe an end-to-end system based on DeepDive that can extract relations from Wikipedia to be ingested in Wikidata with high precision. So far, our approach was applied to five types of relations and identified over 140k relations to be added to the knowledge base.

This paper is organized as follows: first, we review the related work and give a general overview of DeepDive. Second, starting from the data preprocessing, we detail the general methodology used. Then, we detail two applications that follow this pipeline along with their specific challenges and solutions. Finally, we report the results of these applications and discuss the next steps to continue populating Wikidata and improve the current system to extract more relations with a high precision.

Background & Related Work

Until recently, populating the large knowledge bases relied on direct contributions from human volunteers as well as integration of existing repositories such as Wikipedia info boxes. These methods are limited by the available structured data and by human power. To overcome this limitation and integrate various online data sources at scale, Knowledge Vault (KV) (Dong et al. 2014) has in particular been proposed. This method allows the construction of a web-scale probabilistic knowledge base by combining facts various data sources including text, HTML tables and HTML trees. To solve conflicts and assess the confidence in each of the extracted fact, KV utilizes its prior knowledge making it more robust over time. The alternative approach proposed in this paper focuses solely on high quality text extractions. On the long run, a similar fusion approach can be used to further improve the precision and extend the number of relations covered.

DeepDive(Zhang 2015) is a new type of data analysis system that provides a full framework to extract knowledge graph relations from raw text. It enables developers to extract structured relations from text and can achieve very high quality, beating human volunteers in highly specific tasks (Shin et al. 2015). Developers define extractors for mentions, candidates and features, and use distant supervision(Mintz et al. 2009) to extract positive and negative training examples. Based on inference rules provided by the user, DeepDive constructs a factor graph model, learns the weights, and performs inference to find the expectation for each relation.

Recently, DeepDive was successfully applied to various relation extraction problems. In the field of paleontol-

ogy, it was used to help geoscientists aggregate information from the geological science literature in a comprehensive database (Zhang et al. 2013). PaleoDeepDive, as the system is called, performed comparably to human readers in complex data extraction and inference tasks. In an entirely separate field, recent work (Mallory et al. 2015) shows successful application of the same framework on a gene interaction discovery application. By crawling a large collection of gene-related literature, the resulting system was able to extract thousands of relations automatically, populating a large and comprehensive database.

Methodology

An overview of our methodology is shown in Figure 1. The pipeline based on DeepDive framework relies on: (1) data preprocessing, (2) candidate extraction, (3) distant supervision, (4) learning, (5) inference and (6) calibration. Note the conditional a feedback loop in this process illustrated by a dashed arrow. After each iteration, we perform an error analysis to detect erroneous patterns and add features or rules to correct them.

Data Preprocessing To extract the knowledge graph relations at a large scale, we used an original dump of English Wikipedia as of February 2015. These large data sets were parsed using the Stanford NLP parser (Chen and Manning 2014). After this preprocessing step, they were annotated with Name Entity Recognition (NER) and Part Of Speech (POS) tags as well as dependency paths. When it is possible, we also perform filtering to keep only relevant pages and reduce the size of the input data. For instance, when extracting family relationships, we can rule out Wikipedia pages about languages or animals using the predefined Wikipedia categories.

Candidate Extraction The extractors take the text corpus and return a set of candidate relations. A candidate relation consists of two entities and the relation we are extracting. For instance, if the relation we want to extract links a person to another by a family link, we consider co-occurring people in a sentence as candidates for the relation. Therefore, we extract them along with a set of features that provide textual information about them from the corresponding sentence. In the current stage of the pipeline, the data set is split by sentences without document-level coreferencing.

Distant Supervision Following what has become standard practice in DeepDive applications, we use distant supervision (Mintz et al. 2009) to apply labels to our candidate target relations. The concept of distant supervision allows us to apply a label, i.e., True, False or Unknown, to every candidate relation in our set of candidates. Once a subset of candidates is labeled, DeepDive can perform the learning and inference process.

Distant supervision requires a set of positive and negative examples. To collect positive examples, we use the crowd-sourced online database Freebase. However, obtaining negative examples is more complex and requires more application-specific knowledge because we need negative examples that are also extracted as candidates by our application.
for example, suppose we are looking to extract a (company, founder) relation from the corpus. A potentially good negative example would be (Google, Eric Schmidt) since it is likely that Google and Eric Schmidt appear in the same sentence and are therefore considered as candidates by our extractors. Assuming that we know that Eric Schmidt is not the founder of Google, this example would be a relevant negative example. See the "Applications" section for more detail on how we build negative examples for the different applications.

Once the set of positive and negative examples is collected, we can use them to label the set of candidates and feed them to the learning engine of DeepDive.

Feature Extraction & Learning DeepDive utilizes factor graphs to compute the marginal probabilities of the candidates being true relation mentions. In order to learn the weights of features and factors in the factor graph, we input the distantly supervised training set to the learning process. The learning engine extracts features from the tagged examples using ddlib, the native DeepDive library, and user-specified features. Ddlib is a large library that extracts common NLP features such as dependency paths or N-grams. It usually allows the initial application to reach fair results that are improved by domain-specific features defined by users.

Inference Once the graph is learned, the inference engine performs Gibbs sampling to estimate the probability that a candidate relation is correct. Following previous work (Niu et al. 2012; Zhang et al. 2013), we set the cut-off probability to 0.9, which means that a candidate relation is considered correct if DeepDive assigns a probability superior to 0.9; other candidate relations are marked as incorrect.

Calibration & Error Analysis We assess the performance of the system at a given iteration through the two common metrics: precision and recall. To evaluate precision, we randomly sample a set of candidate relations predicted with a confidence above 0.9 by DeepDive and manually label them to establish the number of false and true positives in the set. To summarize, we define C as the set of candidate relations and $E_{dd}(c)$ as the expectation of c outputted by DeepDive for $c \in C$:

$$\#\textbf{(TP)} = \sum_{c \in C} I(c = True \cap E_{dd}(c) \geq 0.9) \quad (1)$$

$$\#\textbf{(FP)} = \sum_{c \in C} I(c = False \cap E_{dd}(c) \geq 0.9) \quad (2)$$

$$\textbf{Precision} = \frac{\#\textbf{(TP)}}{\#\textbf{(TP)} + \#\textbf{(FP)}} \quad (3)$$

Evaluating recall is more tedious than precision. Instead of sampling candidate relations where there is a high expectation like we did for precision, we sample a larger number of candidate relations independently of what DeepDive predicts. We manually label this random sample and compute the recall using the following equations.

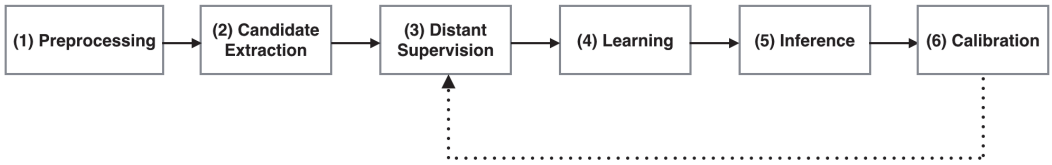

Figure 1: Application pipeline illustrating the DeepDive methodology.

$$\#\textbf{(TP)} = \sum_{c \in C} I(c = True \cap E_{dd}(c) \geq 0.9) \quad (4)$$

$$\#\textbf{(FN)} = \sum_{c \in C} I(c = True \cap E_{dd}(c) < 0.9) \quad (5)$$

$$\textbf{Recall} = \frac{\#\textbf{(TP)}}{\#\textbf{(TP)} + \#\textbf{(FN)}} \quad (6)$$

Aside from giving us an overview of the system's performance, this step allows us to analyze errors, which in turn helps us detect common patterns. Using MindTagger (Shin, Ré, and Cafarella 2015), we can visualize every candidate relation in its original sentence along with the expectation inferred by DeepDive. Once we detect a sentence of interest (false positive or false negative in general), we closely look at the cause of misclassification (Shin et al. 2015) and note the most common patterns of errors. We then engineer more features or distant supervision rules to overcome these general mistakes. We also keep the manually labeled examples in a holdout set that will only be used for final evaluation.

Applications

Populating a knowledge graph of the size of Wikidata is a tremendous task. Therefore, we focus here on a few relations to prove the feasibility of our approach. We started with the relation $(company, founder)$. Then, we moved to the ambitious and large-scale project of family tree including spouses, parents, children and siblings. In this section, we detail these applications one by one, focusing on the pipelines and specific tasks performed.

Founder

The first application we aimed at extracting was company-founder relations directly from text. The choice of this relation was based on two factors: first, it is a static relation that doesn't change over time. Second, it was a consequent problem on Wikidata: our investigation revealed that there are 44301 English Wikipedia company articles that don't have founder information in their Wikidata item and only 2078 that do.

Following the methodology described above, after regular preprocessing, we filter the dataset keeping only documents that are likely to contain information about businesses such as Wikipedia business-related pages. Then, we can extract candidate pairs $(company, founder)$ by extracting all co-occurrences of a person and an organization in the same sentence. We detect these occurrences using NER tags.

For the distant supervision step, we start by collecting a set of ground truth positive examples using Freebase API. We tag all positive examples as true relations if they are extracted as candidates, independently of the content of the sentence they were extracted from. For instance, since the founder relation Bill Gates-Microsoft is in Freebase, we tag as true all candidates relations Bill Gates-Microsoft extracted from Wikipedia corpus.

Then, we generate a set of negative example as follows: for each company in our positive set, we collect its board members and executives that are not part of founding team and tag them as negative examples. This approach outputs a set of relevant negative examples since executives or board members are likely to appear in the same sentence as the corresponding company without necessarily being the founders.

To illustrate this approach, let us review our Google example more specifically. By querying Freebase, we obtain two positive related examples (Google, Larry Page) and (Google, Sergey Brin). To obtain negative examples, we query Freebase for all the executives and board members of Google, such as Eric Schmidt. Assuming that the first query retrieves all the founders, we can infer that all the retrieved entities except Page and Brin are negative. This assumption was empirically validated: we discovered that companies with founder information tend to contain the full founding team. Using the set of positive and negative examples, we can use distant supervision to tag all the labeled candidates. One can notice that a sentence may contain Larry Page, Eric Schmidt and Google. We extract here two candidates: (Google, Eric Schmidt), labeled negatively, and (Google, Larry Page), labeled positively, each with their own set of features.

Using these examples, we can proceed to the iterative development of features and inference based on error analysis. Note that, in this application, the factor graph is a simple logistic regression model because we are considering a single variable.

Family Tree

A second application of this pipeline consists in constructing a family tree of the people in Wikipedia. We focused on four specific relations: parent, sibling, child and spouse. Each of the four relations is with respect to the person that the corresponding Wikipedia page is about. For example, if the page is about Tom Cruise, using the sentences in that

page we only extract the children, parents, siblings and spouses of Tom Cruise. We do so to simplify the process of entity linking. For instance, if the name Tom is mentioned in Tom Cruise's page, it is highly likely that it is referring to Tom Cruise. Relaxing this assumption and generalizing the knowledge search to all pages introduces complexity because entity linking and pronoun-entity linking becomes much harder when we don't know who the page is about. Moreover, we observed that considering all pages, and not only pages of people or company, did not really improve recall. Indeed, it is very rare that a page provides interesting information about a person while a specific page for this person doesn't exist. Therefore, limiting the pipeline to pages of people and companies is very reasonable.

The pipeline for the family tree application is similar to the pipeline we used for the company-founder application, and as previously explained, it consists of mention, candidate, and feature extraction, distant supervision, building the factor graph, learning, and inference. However, in the family tree application, we want to extract each of the relations (spouse, sibling, parent and child) simultaneously and build the interdependencies of the relations into the model. Therefore, the factor graph model used in the family tree application is more expressive and requires further explanation.

In the family tree application, these four relations correspond to respective variables in the factor graph model and the expectation for each of the variables is inferred through marginal inference. Each candidate, i.e., sentence containing two people mentions, gets assigned four variables (sibling, spouse, parent, child). These variables are each connected to multiple $is\_correct$ factors, one for each feature. The $is\_correct$ factor weights depend on the corresponding features and are learned in the learning phase by DeepDive.

In addition to the $is\_correct$ factors, each pair of variables is connected to a MutualExclusion factor. This factor is an or construct, Or(!A, !B), which evaluates to false only if both A and B evaluate to true. This encodes the assumption that if two people have a family relation of type A, they can't have a family relation of type B. For example, child and parent can't be siblings. The weights for these factors are set beforehand to be very large to encode the domain knowledge in the model. A visual representation of the factor graph can be seen in Figure 2.

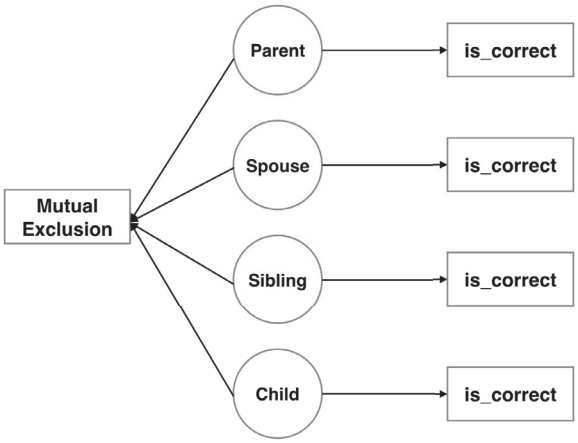

Figure 2: Illustration of factor graph of the family tree application. It includes the various relations we are predicting as well as the mutual exclusion factor between the family relations.

Implementation Challenges

Working on real data comes with various implementation challenges. In this section, we detail the most important ones and the solutions used for our applications.

Parsing Issues

Parsing issues are ubiquitous in NLP. In order to identify the patterns and solve them, we used an error analysis approach (Shin, Ré, and Cafarella 2015). Common patterns include parenthesis or special characters in the text that hinder our ability to extract full names by adding "Null" NER tags between "Person" tags.

To solve these issues, we built simple heuristics. For instance, ignoring special characters after a "Person" tag helps to resolve the parenthesis issue. Also, at the beginning of a Wikipedia page, the subject name is usually repeated twice, once in English and once in the original language. We also solved this by detecting pattern of duplication of identical or very similar names. In general, solving these minor errors allows a considerable boost in performance.

Coreferencing heuristics

Another common pattern was that Wikipedia contributors tend to use only a first or last name when discussing personal matters. For instance, on the Bill Gates page, we can read that Gates married Melinda French in January 1st, 1994. Since the end goal of the project is to push back results to Wikidata, it is important to extract relations between full entities, not partial names. However, we noticed that most of these relations can be extracted by simple rules and heuristics. Therefore, we deployed a simplistic, yet efficient, coreferencing system that allowed us to link these partial names to their full entities. We used a heuristic that tries to match up partial names to the full name of the page when studying a person. Then, if we identify a relation, say parents, we complete the name of the second person as well if needed. This fulfills the requirement of retrieving full entities.

Another issue was coreferencing sentences such as "He married Melinda French," and link the pronouns to actual entities as well. To do so, we used heuristics that links the pronoun to a noun in the previous sentence (when there is

Feature	Weight
No dependency path detected between mentions	-2.04
(Inverse dependency path: $prep\_of \rightarrow nsubj$, 1-gram: father)	.95
Number of people between > 2	-0.89
(Inverse dependency path: poss, 1-gram son)	0.80

Table 1: high performing features of the child relation

only one name in the previous sentence and no other pronoun for instance) and then link all the pronouns "he" in the following sentences. Similar heuristics allow us to extract an additional 1 million people mentions.

Feature engineering

In the presented applications, we began by using the generic feature library (ddlib) provided by DeepDive that enabled us to extract, among others, dependency path features, N-gram features and POS tag features(Zhang et al. 2014). Using all of the features provided by ddlib resulted in a model with relatively high-recall but quite low precision. The model tended to overfit and many seemingly irrelevant features were given a high weight due to their prevalence in the text.

Naturally, the next step in the feature engineering process consists of reducing the feature space from all of the features provided by the generic feature library to a highly relevant subset. This was done through error analysis using the error analysis tool Mindtagger and through manually observing the features that had high learned weights. The set of relevant features was dependent on the application and the relation, but we noticed that, unsurprisingly, dependency path features provided the highest precision but required a large training set.

After narrowing the features into a subset of the generic features, we analyzed the precision and recall errors we made and proceeded to implement features more specific to the application. These features included bucketing existing features in order to compensate for a small training set, cross-products of existing features (dependency path and specific n-grams), and numbers of people, words or organizations inside the relation. Table 1 presents high-performing features of the child relation in the family tree application as an example.

Results

Before discussing the results of our applications, it is important to understand the output of DeepDive. Once the system is trained, and candidates extracted as described in the pipeline, every prediction happens with a certain confidence or probability and it is up to the developer to fix a threshold for the relation to be evaluated as true. In our case, following DeepDive convention, we fixed this threshold to 0.9 and evaluated our precision and recall based on that. Figure 3 shows the resulting precision for the extraction of the spouse relationship versus the confidence on the test set in the left graph and the number of predictions for each bucket on the right graph. First thing to note about the number of predictions in the buckets is the U-shape. This shows that the trained model output either a very high confidence to be true (above 0.9) or a very small one (below 0.5) indicating that it is quite confident about its predictions. This is the first sign that precision is high for this relation.

The results obtained for these applications are summarized in table 2. As we can see from the table, the recall of our extraction is not very high. The reason for that lies on the tradeoff between recall and precision. Since the end goal of our project is to push data back to Wikidata, we tailored our models to have very high precision with a reasonable recall.

Some relations are trickier than others. For instance, in our family tree application, sibling relation seems to be easier to catch than a parent relation. The gap can be even broader between other sets of complex relations. Extracting these relations using our methodology can be further enhanced using human verification. Tools such as on https://tools.wmflabs.org/wikidata-game/distributed/ allows the community to perform this verification through games while also providing us more positive and negative examples. This new input data can also be used for more error analysis to further improve the system.

Discussion and next steps

To conclude, this paper presents a detailed methodology based on DeepDive to extract facts and relation from the Wikipedia corpus with high precision and recall. By building relevant training examples, extractors, features and factor graph models and dealing with scaling and parsing issues, we can now expand this work to more relations, automatically populating Wikidata with all sorts of new facts.

We are currently pushing these relations to Wikidata and wrote a meta page for this project. Working with the DeepDive team, we are implementing tools to get feedback from the Wikidata community about the data and potential next leads.

To continue the Wikidata KBC, future steps include:

- Improve the text chunking by building a whole DeepDive process for good pronoun-entity linking. In particular find a significant number of negative training examples and relevant features. That would drastically improve our recall.

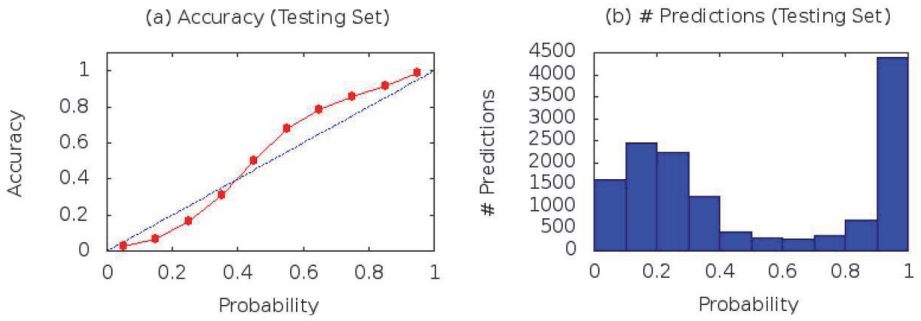

Figure 3: Accuracy and number of predictions of the spouse relationship versus the level of confidence.

Relation	Precision	Recall	# of relations extracted
Company-Founder	88%	50%	4800
Parent	87%	37%	57960
Spouse	90%	52%	58510
Sibling	98%	70%	20674
Child	96%	16%	2516

Table 2: precision, recall and number of relations extracted

- Work closely with the Wikidata team to automate the integration of the extracted relations in the knowledge base.
- Expand to more relations. In particular to interdependent relations for which our current results would help us build a relevant factor graph.
- Make the learning system online to learn automatically from community feedback and autonomously improve over time.

Acknowledgement

We would like to thank Leila Zia for her mentorship throughout the project and for making our project impactful within the Wikimedia Foundation. We would also like to thank Jaeho Shin, Raphael Hoffmann, Ce Zhang, Alex Ratner and Zifei Shan of the DeepDive team for their help and advice regarding DeepDive.

References

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web* 7(3):154–165.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka Jr, E. R.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, 3.

Chen, D., and Manning, C. D. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, 740–750.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 601–610. ACM.

Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J.; et al. 2010. Building watson: An overview of the deepqa project. *AI magazine* 31(3):59–79.

Kasneci, G.; Ramanath, M.; Suchanek, F.; and Weikum, G. 2009. The yago-naga approach to knowledge discovery. *ACM SIGMOD Record* 37(4):41–47.

Lao, N.; Mitchell, T.; and Cohen, W. W. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 529–539. Association for Computational Linguistics.

Mallory, E. K.; Zhang, C.; Ré, C.; and Altman, R. B. 2015. Large-scale extraction of gene interactions from full-text literature using deepdive. *Bioinformatics* btv476.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, 1003–1011. Association for Computational Linguistics.

Niu, F.; Zhang, C.; Ré, C.; and Shavlik, J. W. 2012. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS* 12:25–28.

Shin, J.; Wu, S.; Wang, F.; De Sa, C.; Zhang, C.; and Ré, C. 2015. Incremental knowledge base construction using deep-

dive. *Proceedings of the VLDB Endowment* 8(11):1310–1321.

Shin, J.; Ré, C.; and Cafarella, M. 2015. Mindtagger: a demonstration of data labeling in knowledge base construction. *Proceedings of the VLDB Endowment* 8(12):1920–1923.

Vrandečić, D. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference companion on World Wide Web*, 1063–1064. ACM.

Weikum, G., and Theobald, M. 2010. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 65–76. ACM.

Zhang, C.; Govindaraju, V.; Borchardt, J.; Foltz, T.; Ré, C.; and Peters, S. 2013. Geodeepdive: statistical inference using familiar data-processing languages. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 993–996. ACM.

Zhang, C.; Ré, C.; Sadeghian, A. A.; Shan, Z.; Shin, J.; Wang, F.; and Wu, S. 2014. Feature engineering for knowledge base construction. *arXiv preprint arXiv:1407.6439*.

Zhang, C. 2015. *DeepDive: A Data Management System for Automatic Knowledge Base Construction*. Ph.D. Dissertation, UW-Madison.

Zhu, J.; Nie, Z.; Liu, X.; Zhang, B.; and Wen, J.-R. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th international conference on World wide web*, 101–110. ACM.