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.

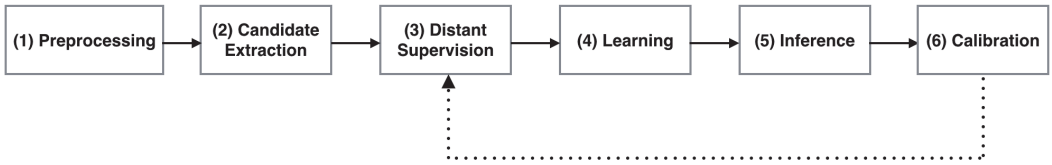


Figure 1: Application pipeline illustrating the DeepDive methodology.

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

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

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

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.

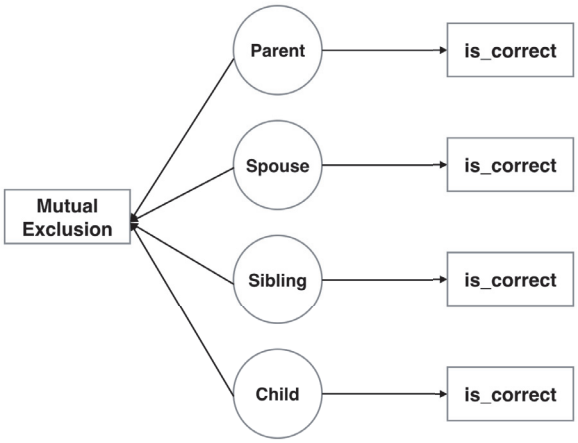


Figure 2: Illustration of factor graph of the family tree application. It includes the various relations we are predicting as well as the mutual exclusion factor between the family relations.

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.

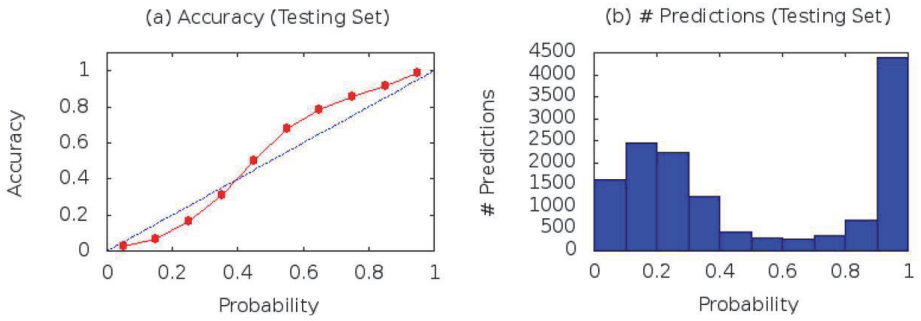


Figure 3: Accuracy and number of predictions of the spouse relationship versus the level of confidence.

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.

材料的疲劳断裂失效预测与预防

孟航程 研 1905 班

学号：2111902118

摘要：对于机械结构零件而言，随着设计过程中不断使用高强度的结构材料来制造承力结构，疲劳断裂事故的发生概率也大大增加。因此，总结疲劳断裂的失效具体特征，分析起影响因素，探讨疲劳失效的预防措施一直都是十分重要的课题。本文主要探究了疲劳失效的特征，分析疲劳失效的影响因素，预测方法，介绍了预防疲劳断裂的若干种方法。

关键词：金属材料，疲劳失效，断裂

一． 引言

疲劳断裂是金属构件断裂的主要形式之一，在金属构件疲劳断裂失效分析基础上形成和发展了疲劳学科。自从 Wohler 的经典疲劳著作发表以来，人们充分地研究了不同材料在各种不同载荷和环境条件下试验时的疲劳性能。尽管大多数的工程技术人员和设计人员注重疲劳问题的预防，而且已积累了大量的试验数据，目前仍然有许多设备和机器发生疲劳断裂。材料疲劳性能的研究，不仅涉及材料的发展和设计水平的提高，而且事关产品和构件的安全使用。

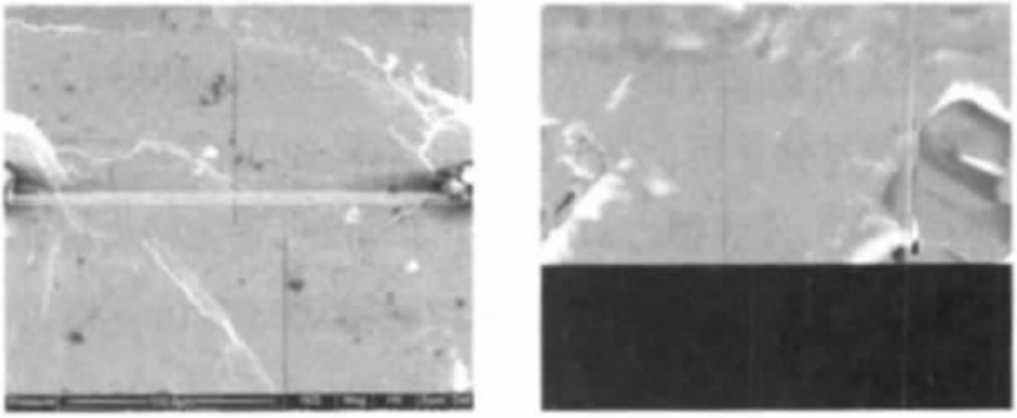
疲劳断裂危害在各种失效形式中最大，数量最多。根据美国国家标准局(NBS)向国会提出的报告，美国由于断裂和为防止断裂所花经费每年竟高达 1140 亿美元，相当于国民经济总产值的 4%。我国在石油、煤炭、铁道、航空和电力等部门都发生过一些恶性事故，损失是巨大的。例如，我国第一架歼八型全天候歼击机和一艘导弹核潜艇在试车时，都曾因疲劳断裂而造成重大损失。如果对材料和结构在使用条件下的行为有深刻的研究和清晰的了解，就能提高产品的设计水平和质量，提高产品的可靠性和使用寿命，防止隐患，那么大部分事故是可以预防和避免的,其经济效益比新增加产品更显著。有报告指出，美国仅断裂分析一项，每年可节约几亿美元,如果通过失效分析使国内发电设备利用率提高 1%，就可制造 30~40 亿元的产值。

二． 疲劳失效特征

疲劳失效是材料在循环载荷作用下发生的损伤和破坏过程。一般而言疲劳断裂包括裂纹的萌生、裂纹的扩展和最终的断裂三个过程，因此疲劳断口上有三个相对应的区域，即裂纹源区、裂纹扩展区和瞬断区。根据所受载荷的水平、材料的力学特性、试样的形状尺寸与约束条件的不同，这三个区域的大小、形状和分布特征也不尽相同，但总体而言可归纳为下列的 4 个宏观规律特征：

（1）疲劳失效为低应力长时间无明显塑性变形的宏观脆性断裂。

（2）疲劳失效是由材料局部的组织不断发生损伤变化并且逐渐累积而成，疲劳总是从最薄弱的区域开始 （见图 1）。



（a）Al2O3 夹杂　　（b）TiN 夹杂

图 1 疲劳裂纹萌生于内部的夹杂物缺陷

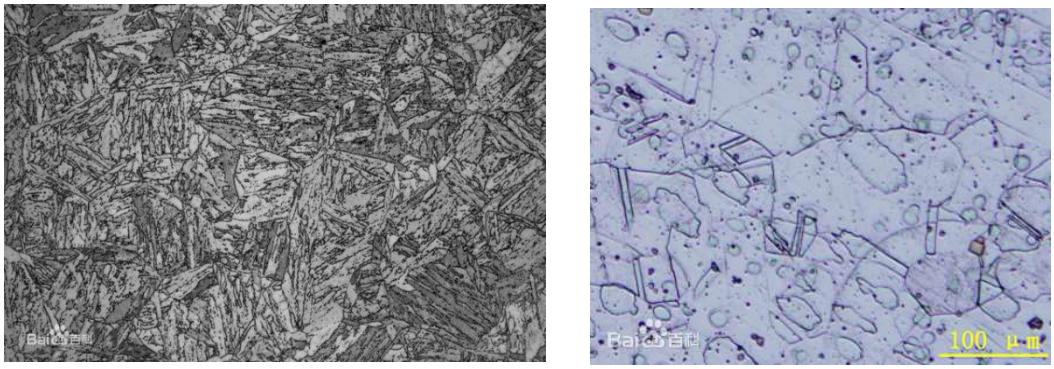
（3）疲劳断裂必须在循环应力和微观局部发生塑性变形，以及拉伸应力作用下发生。前者是裂纹形成的条件，后者是裂纹扩展的需要。

（4）具有随机性，裂纹的形成与扩展都需要一定的晶体学条件、力学条件和变形的协调条件，而且材料本身的组织结构、成分偏析与夹杂缺陷等的不均匀性，决定了疲劳失效具有随机性。

三． 影响疲劳性能的主要因素

影响疲劳性能的因素很多，但对疲劳性能影响最大的主要因素有：

（1）组织结构 一种材料的成分虽然相同，但如果采用不同的热处理方法，则可形成不同的组织结构，如奥氏体和马氏体，这将导致不同的疲劳性能。一些材料采用表面激光淬火来得到表层马氏体和心部奥氏体的组织（如图 2 所示），利用表层的高屈服强度和心部韧性，以综合提高构件的疲劳性能。



（a）马氏体组织　　（b）奥氏体组织

图 2 马氏体与奥氏体组织

（2）表面状态 疲劳失效常从表面开始，一方面这与表面往往承受最大的交变载荷有关；另一方面表面易于存在加工缺陷或与外界接触产生局部的缺陷。不同的表面处理可产生不同的疲劳性能，如采用表面强化（喷丸、滚压和挤压等）可显著改善疲劳性能，而脱碳、电镀和阳极化等处理可降低材料或构件的疲劳性能（见图 2）。不同的机械加工状态导致的表面粗糙度和残余应力不同，也对疲劳性能影响很大。表面状态主要包括表面粗糙度、表层残余应力等，有时也被称为表面质量。

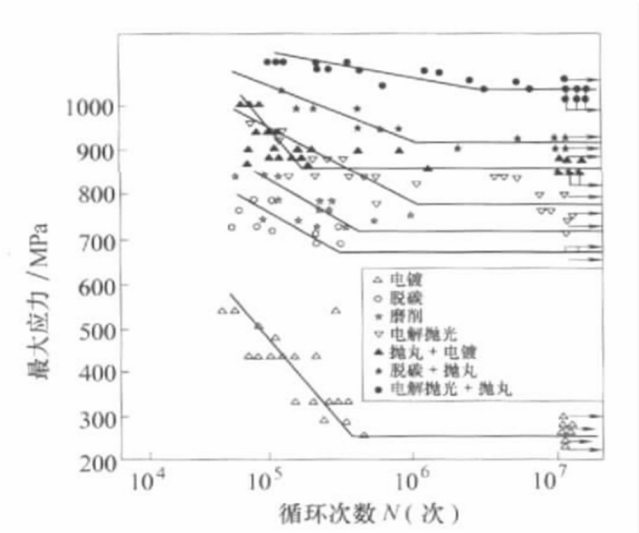


图 3　40CrNi2Si2MoVA 钢不同表面状态下的疲劳性能

（3）应力集中 缺口、圆角或其他几何形状的突然变化都将导致应力集中，从而降低材料的疲劳性能，因此在设计零件时应注意采用适当半径的圆角逐步进行过渡，以确保危险截面的承载能力。此外，不同材料对应力集中的敏感性不同，一般高强度材料对应力集中比较敏感，因此在选用高强度材料进行替代时，应注

意高强度材料的应力集中敏感性，并采用更大半径的圆角进行逐步过渡。

（4）试验条件 试验的频率、温度、应力比和载荷 类型等因素对疲劳性能具有一定的影响。加载频率对疲劳性能的影响随材料的不同而有所差异。温度升高一般可使材料的疲劳性能下降，但使应力集中的影响减小。对于给定的最大载荷，应力比或平均应力越小，疲劳强度越低。

（5）周围环境 在腐蚀环境的作用下，材料的疲劳强度明显降低，材料的疲劳寿命显著缩短。这一方面与疲劳裂纹常从局部的腐蚀坑开始有关；另一方面腐蚀环境常促进新表面的形成，裂纹易于形成和扩展。

四． 疲劳失效预测方法

（1）名义应力法

基本假设：对任一构件（或结构细节或元件），只要应力集中系数 KT 相同，载荷谱相同，它们的寿命就相同。此法中名义应力为控制参数。名义应力法是建立在 S-N 曲线和 Palmgren-Miner 线性累积损伤准则基础上的，也称为总寿命法。该方法应用十分广泛，目前各国的桥梁设计规范中大部分都仍然沿用此方法，如英国 BS5400 规范、美国 AREA 铁路钢桥规范、欧洲标准协会钢结构设计规范及我国铁路桥梁钢结构设计规范，均应用此方法进行疲劳失效预测。

（2）局部应力应变法

基本假定：若一个构件的危险部位（点）的应力-应变历程与一个光滑试件的应力-应变历程相同，则寿命相同。此法中的局部应力-应变是控制参数。

（3）场强法

一些科研人员认为：应力场强法在缺口根部存在一破坏区，它只与材料性能有关，并提出一个描述构件受载严重程度的控制参数—应力场强度σ_{FI}，它综合地反映材料性能、缺口根部的最大应力、缺口附近的应力梯度和应力状态等参数。应力场强法的基本假设是：相同材料制成的构件（元件或结构细节）如果在疲劳失效区域承受相同应力场强度历程，则具有相同疲劳寿命。此法的控制参量是应力场强度。

（4）能量法

基本假定：由相同的材料制成的构件（元件或结构细节），如果在疲劳危险区承受相同的局部应变能历程，则它们具有相同的 疲劳裂纹形成寿命。能量法

的材料性能数据主要是材料的循环应力—应变曲线和循环能耗—寿命曲线。

五． 疲劳失效预防措施

从疲劳失效的预防来看， 预防措施主要有以下几个方面：

（1） 合理的结构设计

对零件进行设计时，设计人员应该加强对疲劳结构的细节设计，要采用合理的结构形状和适当的过渡区域，来保证零件关键部位所受应力低于疲劳抗力，并给予一定的安全设计系数来考虑疲劳失效的概率。在设计阶段不仅应注意构件局部的应力集中，更应注意构件所采用的材料特性，因为材料不同，其疲劳的应力集中敏感性也相差很大。

（2） 合适的材料选择

零件的形状尺寸相同，采用不同的材料来加工制造，其疲劳性能将截然不同。典型的例子是对于飞机起落架，虽然形状相同，但采用强度不同的 GC4，30CrMnSiNi2A 和 300M 来加工制造，其飞行寿命相差很远。对不同使用条件下的零件应选择性能相适应的材料，在室温下选择晶粒细小、强度高和韧性好的结构材料比较耐疲劳；而在高温下选择晶粒适当、组织稳定、强韧性配合较好的结构材料才可提高疲劳寿命。合适的材料是制造长寿命抗疲劳零件的基础。

（3） 适宜的热处理方法

选择了一定的形状尺寸和合适的材料，对材料进行适宜的热处理显得非常关键。相同的材料，热处理制度不同，疲劳性能也就完全不同。40Cr 材料在不同温度下进行回火或对其进行渗氮化学热处理，其疲劳强度绝对数值可差 500MPa，相对数值差为 40%。热处理时不仅需要关注材料热处理后的组织与性能，更需要保证热处理制度的稳定性、可靠性，应避免在工件表面形成脱碳、过热、硬脆相等变质层，以确保表面层的疲劳抗力。

（4） 可靠的零件加工工艺

零件的加工质量对其使用寿命的影响很大，相同的材料即使采用相同的热处理制度，但如果选用不同的加工方式进行制造，其疲劳性能也相差很大，尤其是对于高强度结构材料。可靠稳定的零件加工制造工艺是零件表层质量的保证，没有好的零件加工工艺就得不到好的表面质量，也谈不上高的疲劳性能。零件加工时要杜绝烧伤、吃刀等工艺所产生的缺陷，并尽量减小加工过程中产生的残余拉

应力数值。

（5） 适当的表面强化处理

表面强化工艺技术虽然起源于结构的维护修理，但近年来已被设计单位和工厂作为一种设计技术和制造技术，用于零件的加工、制造与维护等过程。零件的设计是基础和依据，零件的材料选择是关键的性能支撑，制造工艺是实现加工的手段和可靠性的根本保证，表面强化则是零件表层性能的守护神。采用合适的表面强化工艺技术处理零件，可使零件不在表面发生疲劳失效，而在表层下的基体处萌生疲劳裂纹，这将充分发挥材料的强度潜力，节省材料的消耗。最常用的表面强化技术为喷丸，喷丸所产生的表面完整性变化如图 3 所示。

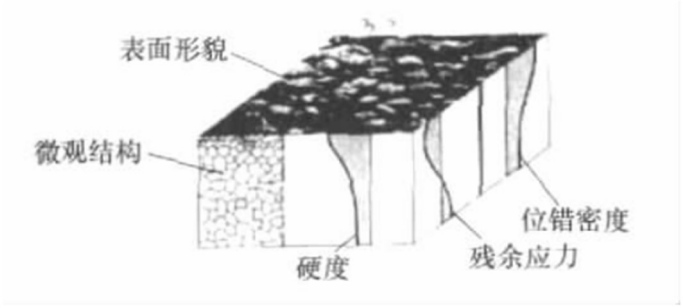


图 4 喷丸产生的表面变化

六． 结束语

疫情无情人有情，疫情期间在家中的这一段学习时间让我无法忘怀。尤其是材料失效分析课，老师新颖的教学方式让我耳目一新。同学们争相发帖、回复、讨论的场景更是前所未有。在这节课程上，不仅让我学到了材料失效分析的基本知识，还让我深刻的认识到，知识只有在交流和碰撞中才能散发它的魅力，我们每一个人都在探索和分享中有所积累。当我思考老师抛出的种种问题时，我越发能够体验到这门课程的奥妙。失效分析是当代工业发展的一根命脉，综合了各个领域的多种专业知识，是一门需要日积月累经验的深度学科，也是关系到生命财产安全的重要学科。

参考文献：

[1] Mcdiarmid D L. A GENERAL CRITERION FOR HIGH CYCLE MULTIAXIAL FATIGUE FAILURE[J]. Fatigue & Fracture of Engineering Materials & Structures, 1991, 14(4).
[2] Yarin A L, Lee M W, An S, et al. Failure, Cracks, Fracture, Fatigue, Delamination, Adhesion, and Cohesion[M]. 2019.
[3] 王旭亮. 不确定性疲劳寿命预测方法研究[D]. 南京航空航天大学, 2009.
[4] 赵子豪, 刘德刚. 焊接结构疲劳失效的产生原因与预防措施[J]. 现代机械, 2012(01): 68-71.
[5] 庄东汉. 材料失效分析[M]. 华东理工大学出版社, 2009.
[6] 高玉魁. 疲劳断裂失效分析与表面强化预防[J]. 金属加工：热加工, 2008(17): 36-38.
[7] 何柏林, 王斌. 疲劳失效预测的研究现状和发展趋势[J]. 机械设计与制造, 2012(04): 284-286.
[8] 王中光. 疲劳断裂和失效分析[J]. 材料科学与工程学报(01): 22-29.