**University of
Zurich^{UZH}**

**Zurich Open Repository and
Archive**

University of Zurich
University Library
Strickhofstrasse 39
CH-8057 Zurich
www.zora.uzh.ch

Year: 2021

Student Perception of Software Engineering Factors Supported by Machine Learning

Marutschke, Daniel Moritz ; White, Jeremy ; Serdült, Uwe

Abstract: Software engineering education requires training in project management techniques in combination with intercultural communication skills to cooperate effectively in today's software development landscape. This poses a challenge to teach software engineering effectively in undergraduate courses for industry needs. International engineering study programs provide a unique opportunity to experience a software development process with participants from different cultures. For most students, English is not their native language, which reflects many distributed software engineering projects. This study surveyed 28 students from an international undergraduate engineering course in Japan on their perceived importance of software engineering factors and freeform text comments on their success, struggle, and critique within this course. A language model is trained using Word2Vec on the textual data to support the results of the perceived importance of software engineering factors.

DOI: <https://doi.org/10.1109/iceccme52200.2021.9591101>

Posted at the Zurich Open Repository and Archive, University of Zurich

ZORA URL: <https://doi.org/10.5167/uzh-209313>

Conference or Workshop Item

Accepted Version

Originally published at:

Marutschke, Daniel Moritz; White, Jeremy; Serdült, Uwe (2021). Student Perception of Software Engineering Factors Supported by Machine Learning. In: 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Mauritius, Mauritius, 7 October 2021 - 8 October 2021, IEEE.

DOI: <https://doi.org/10.1109/iceccme52200.2021.9591101>

Student Perception of Software Engineering Factors Supported by Machine Learning

Daniel Moritz Marutschke
College of Global Liberal Arts
Ritsumeikan University
Osaka Ibaraki, Japan

moritz@fc.ritsumei.ac.jp, 0000-0001-5847-7440

Jeremy White
College of Information Science and Engineering
Ritsumeikan University
Minami-Kusatsu, Japan

jwhite@fc.ritsumei.ac.jp, 0000-0002-5939-461X

Uwe Serdült
College of Information Science and Engineering
Ritsumeikan University
Minami-Kusatsu, Japan
serdult@fc.ritsumei.ac.jp, 0000-0002-2383-3158

Abstract—Software engineering education requires training in project management techniques in combination with intercultural communication skills to cooperate effectively in today's software development landscape. This poses a challenge to teach software engineering effectively in undergraduate courses for industry needs. International engineering study programs provide a unique opportunity to experience a software development process with participants from different cultures. For most students, English is not their native language, which reflects many distributed software engineering projects. This study surveyed 28 students from an international undergraduate engineering course in Japan on their perceived importance of software engineering factors and freeform text comments on their success, struggle, and critique within this course. A language model is trained using Word2Vec on the textual data to support the results of the perceived importance of software engineering factors.

Index Terms—Software Engineering, Problem-Based Learning, Machine Learning, E-Learning

I. INTRODUCTION

The gap between industry needs and software engineering education has been of interest for both fields since its inception and continues with new concepts and technologies [1]–[3]. A continuing challenge is teaching distributed projects management and intercultural communication in an engineering environment [4]–[6]. Both need educational environments that can be difficult to provide by universities. Students need to learn to work with team members who speak different languages and who come from different cultural backgrounds.

An international course taught in Project-Based Learning (PBL) style and following the Conceive, Design, Implement, and Operate (CDIO) stages was the basis for this study. Students were surveyed in summer of 2019 after completing a working prototype in the context of smart city development and to rate software engineering factors by their perceived importance.

The CDIO stages are part of implementing aspects of a real-world development scenario [7]–[9]. The course spans 15 weeks and has six academic hours per week (3×90 minutes).

Students cover two projects within one semester, a proof of concept (week 1–6) and a prototype development (week 7–15). Topics covered include the software lifecycle and its models, quality management, modern practices and future trends in software development, requirements elicitation, software project management, advanced techniques of software development, i.e., software reuse, software testing and validation, software product documentation, Unified Modeling Language (UML). As the importance of requirements engineering and validation is emphasized by Crawley et al., requirements elicitation and updating specification are performed throughout the semester [9]. A validation and reflection of the requirements of each project is done during one poster presentation for the first project and a prototype demonstration together with a final report for the second project.

Project-based learning (PBL) has been used for more than 25 years and has experienced a comeback in recent years due to the success of student-centric teaching in engineering education [4], [10]–[13]. Students are typically assigned a project as an external outcome, determined by a product owner or customer, and have to self-organize and construct their own solutions to a case study. A lecture-centered instruction-style, passively consuming knowledge is replaced by small autonomous groups with instructors facilitating students' work. Hybrid versions of PBL have been successfully implemented in undergraduate software engineering projects to effectively teach soft skills. Savery differentiates between project-based and problem-based learning, where the former is being externally defined as the desired end product and the latter allowing students to define project specifications intrinsically [14]. Students in the course describes in this paper were assigned to work on a concrete project, consulting with an external field-expert for the second project. For a successful learning outcome, a disciplined planning phase by all instructors, well implemented guidelines, and a high degree of guidance and oversight throughout the project are needed.

The prototype phase was taught in software-reuse process

model. Other methods, such as the software lifecycle, waterfall model, and agile software engineering were covered in theory lessons.

The goal of this paper is to support evaluating quantitative survey results with insight using machine learning on unstructured textual data. As the experience of each student is different, distributions will vary and underlying factors can help in understanding the data variance.

The paper is structured as follows: Section II covers previous work on software engineering, its education, and language models created by Word2Vec. Section III describes the research method, detailing the questionnaire design in Section III-A, usage of Word2Vec in Section III-B, and combination of the two in Section III-C. Results and discussion are covered in Section III. The paper concludes in Section V.

II. PREVIOUS WORK

This section covers related work regarding software development processes influenced by cultural differences and research on the diverse use of Word2Vec by applying this machine learning and natural language processing to create language models.

A. Software Engineering and Intercultural Competency

Research on intercultural communication and collaboration in software development often focuses on global software engineering and distributed teams [15], [16]. Many of the struggles could also be observed in a local development team with members from different cultures, such as language barriers, work and communication styles, and overestimating software tools for communication and development [17]. Other factors, such as timezone differences, asynchronous communication, and the formation of an in-group and an out-group are of no or less concern.

Hofstede et al. analyzed data from thousands of IBM employees worldwide and classified different cultural dimensions that can help make international teams more sensitive to country-specific differences [5].

Hoda et al. stressed the importance of learning to deal with diverse languages, concepts of time, and assumptions about culture [6]. They identified seven dimensions that influenced their courses the most: language differences, concept of time, attitude towards grades, assumptions about national culture, differences in autonomy, influence of the course lecturer, and work habits.

An English speaking environment that many students enter as non-native speakers can hamper communication in several ways. The speed of interacting can be reduced due to mental overhead in converting thoughts from their native language into English. Answers might then need to be translated back from English into their native language. Communicating this way may also increase anxiety in group members [18].

MacGregor et al. underline the importance of increasing awareness of cultural differences to work more efficiently. One of the main challenges in intercultural teams are misunderstandings due to these cultural differences [19].

Brockmann et al. ran machine learning classification algorithms on a large job market database and found that under-represented groups with diverse skills can help to contribute to a broader perspective in computer science [20].

B. Uses of Machine Learning and Word2Vec

In big data analysis and data mining, language models have gained popularity, as they allow novel insights into large, unstructured textual data. The presented researches show only a fraction of the inspired publications regarding Word2Vec and other machine learning algorithms. These approaches, however, also garnered discussions about implicit bias in large models [21]–[23]. Although the problem of implicit bias is not a concern in the small dataset used in this research, it can be one major drawback with growing data. A list of previous publications and research is presented below.

Altszyler et al. investigate small text corpora for word associations and find that Latent Semantic Analysis (LSA) can provide superior results for low-frequency words compared with Word2Vec [24]. They apply their algorithms to dream journals and are focussed on psychological research.

A large corpus of abstracts and papers related to biomedical research was analyzed by Zhu et al. [25]. They conclude that increasingly large datasets can result in more relations in biomedical terms, but do not guarantee better precision. This ties into the strength of word associations, but poor semantic properties of Word2Vec.

Kazama et al. propose an ingredient substitution method based on the vector arithmetic allowed by the Word2Vec algorithm [26]. Their research potentially allows to change cuisine styles from one cultural background to another, but also to adjust meals to a persons's individual dietary preferences. The study also lists limitations on the lack of comprehensive data and traditional dietary pattern being mostly described in its native language.

Other limitations and strengths of Word2Vec were investigated by Di Gennaro et al. [27]. Overfitting was mostly ruled out for the training process of large datasets and they list analogy models as a strength. Syntactic relationships were not able to be achieved by Word2Vec.

To allow comparing future studies and allow context shifting vector arithmetic, the Word2Vec tool was chosen to combine with the box plot analysis in this research.

III. RESEARCH METHOD

This section describes the questionnaire design and how the quantitative results are combined with results from applying machine learning to qualitative textual data. The overall overview of the steps involved are detailed in Fig. 1.

A. Questionnaire Design

An anonymous printed questionnaire was distributed to students after the completion of their final prototype.

Providing a Likert scale ranging from 1 (least important) to 5 (most important), students were asked to rank the following eight software engineering factors by their perceived importance:

Fig. 1. Research overview from questionnaire conduct, box plot analysis, machine learning process, to the combined evaluation.

- Language differences
- Proficiency in shared language
- Cultural differences
- Familiarity between teams
- Trust within team
- Communication between members
- Software and hardware tools
- Leadership

These factors are based on the cultural dimensions extensively investigated by Hofstede et al. and previous research on global software engineering by Marutschke et al. [5], [28]. While previous research also included aspects of time-differences and distributed teams, factors were adopted to fit mostly face-to-face communication and local teams.

In this study, more than half in the undergraduate course are international students, making the course structure and experience unique.

Four freeform questions were included to ask students for their feedback regarding the software engineering class on the following:

- What did students enjoy the most
- What did students struggle with the most
- What did students dislike the most
- Other comments about their experience

B. Language Model with Word2Vec

Machine learning algorithms to create language models can help analyze unstructured textual data by converting words into numerical vector values called *word embeddings*. By training Word2Vec on a text corpus, words can be compared numerically. Similar words, that occur more frequently in the same textual context are close in vector space. The algorithm allows to create a model that returns a numerically close word, given a context, or return a numerically close context, given a word. The former is called continuous bag-of-words (CBOW) and the latter a skip-gram model and used in this research.

As words are represented as multidimensional vectors, the algorithms also allows to perform vector arithmetic [29]. Word embeddings can be added to each other or subtracted from each other, which numerically shifts the predicted context or target word in the word-vector space.

The natural language processing (NLP) toolkit developed by the Stanford NLP group was used to tokenize and lemmatize the words [30]. Stop-words were removed from the sentences and all words were converted into lowercase. This maximizes the training on meaningful words and consolidates words with the same meaning, but either in different grammatical context or with different upper and lower cases. While this can be controlled for in this smaller dataset, it is also a weakness of the Word2Vec algorithm, as it trains on case-sensitive data and does not correct for, e.g., misspellings.

Word2Vec was trained on the dataset with a vector size of 60 to avoid overfitting (observed with smaller datasets). The window size was set to 5 to include five words on the left and five words on the rights of each target word. The minimum word count was set to 1, ensuring that unique words also get processed. This results in a Bernoulli model, predicting the context, providing a particular word, given by $P(w_1 \dots w_m | w)$, where w is a single target word as input and m context words $w_1 \dots w_m$ as output. For higher accuracy, negative sampling is used on contexts [31]. The initial learning rate was set to $\alpha = 0.001$. As the training corpus was small for this experiment, the results could slightly be improved compared to the default $\alpha = 0.025$.

C. Combining Questionnaire Results

With subjective data gathered from undergraduate students in their second year, there are several caveats to consider. Although the subjective data can be numerically represented and statistically plotted, the complex nature of a software engineering process can result in various experiences. Another factor is the lack of experience of students at this level in their education. This can be observed when compared to software development in graduate school and should be taken into account.

To have additional insight in the perceived importance of software engineering factors, machine learning was applied to the freeform texts and its results used to interpret the variance or tendencies of the plotted values.

The freeform text questions were designed to gather positive and negative experiences during the software development process. Students could elaborate on their success, struggle, and criticism they underwent. This also allows to connect different factors or put thoughts into context. Using machine learning to create a language model can help with these otherwise hidden information.

IV. RESULTS AND DISCUSSION

In this study, 28 second-year undergraduate students from an international engineering program at a Japanese university took place. Although detailed information about age and gender were not collected, it should be noted that more

than three quarter of the students were male. Over half of the students moved abroad to Japan for this program, with most students from Asia, and the remaining from Japan. As described in Section III-A, eight software engineering factors were ranked by students according to their perceived importance. The results are displayed as box plots in Fig. 2.

The freeform text questionnaire yielded 101 answers (112 possible answers with 11 empty cells). As the dataset is still small for vectorized machine learning, the text was pre-processed to maximize word frequency and meaning (Section III-B).

Fig. 2. Students' perceived importance of software engineering factors are shown as box plots. Median values are shown on the right of each box plot.

Students use English as shared language in their studies. The university program is also designed with an all-English environment in mind. Students have to have a minimum English proficiency to enter the program, which still results in considerable differences in their ability to communicate. *Language differences* in Fig. 2 shows the biggest amount of variance with the medium split down the middle. This points to a variety of experiences by students in using English and conveying their intent to others.

The factor *Proficiency in shared language* has a slight skew toward higher importance, when compared with *Language differences*.

Cultural differences were considered less important by most, with a few outliers. Especially in an already international environment, this factor might not be perceived as influential. This factor has been observed to be of more importance when cultures cluster in distributed teams [28].

The *Familiarity between teams* shows a tendency towards a factor perceived more important. In undergraduate studies in particular, friendships are being formed and fitting in a group might be considered more important than the study outcome.

As the software development process depends on teamwork, the importance of delegating work and reliable team members becomes apparent. The box plot for *Trust within team* displays that students perceived this factor as important.

From Fig. 2, the most important factor was deemed *Communication between members*, values only containing 5 and 4, with the latter being an outlier. Students had to hold weekly meetings and report on those meetings. Schedules and upcoming tasks were kept with communication boards and a centrally managed online Kanban board.

The factor *Software and hardware tools* had a skew towards being perceived more important. With more experience comes the realization that software tools are only a means to an end, as students from global software engineering in graduate school often report [28].

The role of *Leadership* was also rated with a strong tendency towards being more important. Team roles were predefined, but as students could self organize in these roles. Students with strong leadership tend to take over the role of project manager, which could explain the distribution, with the occasional outlier.

For the language model, the dataset gathered from the freeform text questionnaire was analyzed as a raw corpus data and a primed corpus data. In the primed corpus, each answer from the questionnaire was primed with a keyword to allow easier context shifting by word-vector arithmetic. The three first questions (not including the free comment) were primed with *enjoy*, *struggle*, and *dislike*.

The following keywords were investigated for their closest context (listed in Table I):

- communication
- team
- group
- problem
- leadership
- language
- trust

As context shifts, the following were considered (listed in Table II):

- struggle — dislike
- struggle — enjoy
- enjoy + struggle — dislike

The values describe in Tables I and II are the results by Word2Vec trained in the skip-gram mode, resulting in an m -word context with a similarity score calculated by cosine similarity. Values can range between -1 and $+1$. Considering the small data sample for this experiment, the top three to top five with values larger than 0.20 were taken into account.

Taking the information from Table I into account, supporting context can be seen for additional insight into the software engineering factors. It should also be noted that the context should not be used assuming semantic causation, as the Word2Vec algorithm can only produce similarities based on vector comparison. Especially for small datasets, words can predominantly be in rare context, as shown for *language*, being associated with *debugging* and *programming*, suggesting the connection to programming language rather than spoken language.

The importance of communication was mentioned frequently in the freeform question and after training on the

TABLE I
KEYWORDS AND CONTEXT MODELED BY WORD2VEC

Keyword	Context (Similarity Score)
communication	cooperate (0.39), need (0.34), teammate (0.33) language (0.33), application (0.29)
team	discuss (0.38), report (0.38), working (0.37) success (0.30), thankfully (0.27)
group	ask (0.37), prototype (0.32), familiar (0.31) documentation (0.31)
problem	flow (0.31), workload (0.30), happy (0.28) opinion (0.27), effective (0.25)
leadership	learn (0.40), teamwork (0.32), imbalance (0.29) responsibility (0.27)
language	interdependently (0.34), difficult (0.30), normal (0.30) debugging (0.27), programming (0.26)
trust	keep (0.41), coding (0.33), imbalance (0.33) agree (0.29), organize (0.29)

TABLE II
WORD-VECTOR ARITHMETIC AND CONTEXT MODELED BY WORD2VEC

Word-Vector Arithmetic	Outcome Context (Similarity Score)
struggle – dislike	motivation (0.28), implement (0.24) latex (0.23), distribute (0.23), design (0.21)
struggle – enjoy	give (0.43), develop (0.36), random (0.32)
enjoy + struggle – dislike	simulate (0.23), app (0.23), latex (0.23) person (0.22)

dataset, similarities can be found for *cooperate*, *need*, and *teammate*, underlining the importance further found in Fig 2.

A difference could be found in *team* and *group*, where the algorithm associated the former with discussion, writing reports and working, where the latter was more similar to *ask* and software engineering related tasks, denoted by *prototype* and *documentation*.

The word embedding for *problem* resulted in less interpretable context. This might be due to students writing about difficulties (*flow*, *workload*) followed by success or resolve (*happy*, *effective*).

The importance of *Leadership* in Fig. 2 is supported by learning within a team, *teamwork*, possible *imbalance* and the significance of *responsibility*.

The relevance of *Trust within team* is connected to keeping the project on track and an all hands on deck mentality. This last factor was only interpretable by consulting the original answers, namely “Keeping our own website and making a prototype of the application.”, primed by *enjoy* and “... keep everyone on track can be hard sometimes.” in the free comments.

The vector arithmetic detailed in Table II was only performed on the primed dataset. In larger datasets, this might not be necessary, rather be an overfitting hazard and will be considered for future work.

To *struggle* can have positive and negative aspects. To look into both independently, *struggle – dislike* and *struggle – enjoy* were performed, respectively. On the positive outcome context, *motivation* was highest. $\text{\LaTeX}$ having a steep learning curve, the initial struggle can result in a feeling of accomplish-

ment and is also part of the list. On the negative side, to *give* tasks to team members, who were less motivated was one of the reoccurring struggles.

An emphasis on perceived positivity was compared by calculating *enjoy + struggle – dislike*. As from *struggle – dislike*, $\text{\LaTeX}$ was included in the list. Simulations and working on the app were described as hard but enjoyable and rewarding.

V. CONCLUSIONS

Software engineering education in intercultural environments has many challenges and with changing needs from the industry and software development landscape, research methods need to adopt to gain insight into effective teaching and knowledge transfer.

An additional difficulty in undergraduate software engineering education is the lack of experience by students, resulting in the need for addressing multiple teaching factors simultaneously.

This research proposes a combination a proven questionnaire design to investigate students’ perception of software engineering factors supported by machine learning on freeform text comments. The results show the value of additional information to interpret distributions in box plot results. A language model can provide information about how students operated, struggled, or succeeded. Where the box plots can point out tendencies or have wide variances, word embeddings and context similarities reveal underlying causes.

This method will be investigated in future works for further insights into software engineering education and other questionnaire types in combination with textual data and Word2Vec.

REFERENCES

- [1] K. Beckman, N. Coulter, S. Khajenoori, and N. Mead, “Collaborations: closing the industry-academia gap,” *IEEE Software*, vol. 14, no. 6, pp. 49–57, 1997.
- [2] N. R. Mead, “Software engineering education: How far we’ve come and how far we have to go,” *Journal of Systems and Software*, vol. 82, no. 4, pp. 571–575, 2009, special Issue: Selected papers from the 2008 IEEE Conference on Software Engineering Education and Training (CSEET08).
- [3] W. Wu, A. Tesei, S. Ayer, J. London, Y. Luo, and V. Gunji, “Closing the skills gap: Construction and engineering education using mixed reality – a case study,” in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–5.
- [4] S. Beecham, T. Clear, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, “Preparing Tomorrow’s Software Engineers for Work in a Global Environment,” *IEEE Software*, vol. 34, no. 1, pp. 9–12, 2017.
- [5] G. Hofstede, G. J. Hofstede, and M. Minkov, *Cultures and Organizations: Software of the Mind, 3rd Edition*. McGraw-Hill, 2010.
- [6] R. Hoda, M. Babar, and Y. Shastri, “Socio-Cultural Challenges in Global Software Engineering Education,” *IEEE Transactions on Education*, vol. Issue 99, 2016.
- [7] T. Ding, “Construction and exploration of university software engineering teaching system based on cdio educational concept,” *Frontiers in Educational Research*, vol. 3, no. 9, pp. 44–48, 2020.
- [8] G. Rechistov and A. Plotkin, “Computer engineering educational projects of mipt-intel laboratory in the context of cdio,” in *Proceedings of the 10th International CDIO Conference*. Universitat Politècnica de Catalunya, June 16–19 2014.
- [9] E. F. Crawley and D. R. B. Malmqvist, William A. Lucas, “The cdio syllabus v2.0 an updated statement of goals for engineering education,” in *Proceedings of the 7th International CDIO Conference*, Technical University of Denmark, Copenhagen, June 20–23 2011, pp. 47–83.

- [10] D. Jiang and J. Lin, "Project-Based Learning with Step-Up Method — Take CDIO Abilities Cultivation in Computer Specialty for Example," in *Proceedings of the 8th International CDIO Conference*. Queensland University of Technology, 2012.
- [11] A. Rodrigues and S. Santos, "A Framework for Applying Problem-Based Learning to Computing Education," in *Proceedings of Frontiers in Education Conference (FIE 2016)*. IEEE, 2016.
- [12] J. Olivares-Ceja, B. Gutierrez, A. Kress, J. Staufer, and P. Brockmann, "Project-Based Learning in an International Classroom to Teach Global Software Engineering," in *Proceedings of International Conference on Education and New Learning Technologies (EDULEARN17)*. IATED, 2017.
- [13] A. K. Peters, W. Hussain, A. Cajander, T. Clear, and M. Daniels, "Preparing the Global Software Engineer," in *Proceedings of 10th International Conference on Global Software Engineering (ICGSE 2015)*. IEEE, 2015, pp. 61–70.
- [14] J. R. Savery, "Overview of Problem-based Learning: Definitions and Distinctions," *Interdisciplinary Journal of Problem-Based Learning*, vol. 1, no. 1, pp. 9–20, Feb. 2006.
- [15] S. Schneider, R. Torkar, and T. Gorschek, "Solutions in global software engineering: A systematic literature review," *International Journal of Information Management*, vol. 33, no. 1, pp. 119–132, Feb. 2013.
- [16] I. Bosnić, F. Ciccozzi, I. Crnković, I. Čavrak, E. N. nd R. Mirandola, and M. Žagar, "Managing diversity in distributed software development education—a longitudinal case study," *ACM Trans. Comput. Educ.*, vol. 19, no. 2, Jan. 2019.
- [17] A. Hoffmann and C. Lescher, "Collaboration and intercultural issues on requirements: communication, understanding and softskills (circus)," in *2009 Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills*, 2009, pp. 1–4.
- [18] R. Sangwan, M. Bass, N. Mullick, D. Paulish, and J. Kazmeier, *Global software development handbook*. York, NY: Auerbach Publications, 2007.
- [19] E. MacGregor, Y. Hsieh, and P. Kruchten, "The impact of intercultural factors on global software development," in *Canadian Conference on Electrical and Computer Engineering, 2005.*, 2005, pp. 920–926.
- [20] P. Brockmann, H. Schuhbauer, and A. Hinze, *Balancing the Tension between Digital Technologies and Learning Sciences. Cognition and Exploratory Learning in the Digital Age*. Springer Nature Switzerland, 2021, ch. Diversity as an Advantage: An Analysis of the Demand for Specialized and Social Competencies for STEM Graduates Using Machine Learning.
- [21] M. Nissim, R. van Noord, and R. van der Goot, "Fair Is Better than Sensational: Man Is to Doctor as Woman Is to Doctor," *Computational Linguistics*, vol. 46, no. 2, pp. 487–497, 06 2020.
- [22] A. R. Rodgers and M. Puterbaugh, "Digital badges and library instructional programs: Academic library case study," *Journal of Electronic Resources Librarianship*, vol. 29, no. 4, pp. 236–244, 2017.
- [23] J. Zou and L. Schiebinger, "Ai can be sexist and racist — it's time to make it fair," *Nature*, vol. 559, no. 7714, pp. 324–326, 2018.
- [24] E. Altszyler, S. Ribeiro, M. Sigman, and D. Fernández Slezak, "The interpretation of dream meaning: Resolving ambiguity using latent semantic analysis in a small corpus of text," *Consciousness and Cognition*, vol. 56, pp. 178–187, Nov 2017.
- [25] Y. Zhu, E. Yan, and F. Wang, "Semantic relatedness and similarity of biomedical terms: examining the effects of recency, size, and section of biomedical publications on the performance of word2vec," *BMC Medical Informatics and Decision Making*, vol. 17, no. 1, p. 95, 2017.
- [26] M. Kazama, M. Sugimoto, C. Hosokawa, K. Matsushima, L. R. Varshney, and Y. Ishikawa, "A neural network system for transformation of regional cuisine style," *Frontiers in ICT*, vol. 5, Jul 2018.
- [27] G. Di Gennaro, A. Buonanno, and F. A. N. Palmieri, "Considerations about learning Word2Vec," *The Journal of Supercomputing*, 2021.
- [28] D. M. Marutschke, V. Kryssanov, and P. Brockmann, "Distributed teams in global software engineering education: Project-based approaches in bachelor and master degree classes," *ICIC Express Letters*, vol. 14, no. 2, pp. 153–162, 2020.
- [29] A. Gittens, D. Achlioptas, and M. W. Mahoney, "Skip-gram – Zipf + uniform = vector additivity," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 69–76.
- [30] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020. [Online]. Available: <https://nlp.stanford.edu/pubs/qi2020stanza.pdf>
- [31] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer International Publishing AG, 2018.