# Text Analysis and Retrieval – Project Vademecum

UNIZG FER, Academic Year 2021/2022

Published: March 15, 2022

## 1 About the Project

The student projects are a gist of the Text Analysis and Retrieval (TAR) course. Groups of two or three students work together on a specific TAR problem, following the steps below. Work on the project is meant to contribute to the following course outcomes: (1) use linguistic preprocessing tools, (2) design and implement a text analysis/retrieval system, (3) apply machine learning algorithms to text analysis tasks, (4) evaluate a text analysis/retrieval system, and (5) organize and formulate a system description paper.

### 1.1 Studying Related Work

To start off, you must read the papers recommended in your task's description. As we by and large rely on past competitions in the field of NLP/IR, these papers are mostly survey and system description papers related to these competitions. By reading through these papers, you'll get an idea how others tackled the task and hopefully get a general idea how to tackle the task yourself. Obviously, you are strongly encouraged to read as many papers as possible. To search for related publications, we recommend Google Scholar,[1] a scientific publication search engine. Make sure to collect the references (quote symbol → BibTeX), as you'll have to cite all related work in your project report.

### 1.2 Analyzing the Data

Each problem comes with a manually-compiled dataset provided by the competition organizers. Usually, the first step is to thoroughly examine the given data and get a "feel" for it. Knowing what you work with is crucial as your machine learning models will be trained on this data. This usually includes constructing sensible set of model features, but goes as far as ruling out some models right off the bat.

### 1.3 Implementing the System

A backbone of each project, of course, is the project implementation. Note, however, that this course is not concerned with programming skills, be it in the machine learning domain or in general. You should implement your models all by yourself, but are encouraged to use the readily-available tools and libraries where needed. Nobody expects you to implement your own Support Vector Machine (SVM) from scratch ☺. To make it easier to work on your implementation within a group, we recommend using Git,[2] a distributed version control system. Needless to say, you are *not* constrained to our beloved Python, and may

[1] http://scholar.google.com/
[2] https://git-scm.com/

use whatever programming language you prefer. What makes Python appealing, however, is that there is a lot of freely-available machine learning and TAR libraries out there.[3,4,5] (Mind you, we use it everyday in our research.)

### 1.4 Evaluating the System

Each machine learning system must be *properly* evaluated to determine how efficient it actually is. This means following the standard evaluation practices in machine learning: model selection, cross-validation, statistical significance testing, and so on. Obviously, each TAR task is evaluated differently and therefore you must evaluate your system according to the recommended evaluation metric. Competition organizers usually provide the evaluation script (and test data), which you can seamlessly incorporate in your code. If there's no test data available, please do a proper evaluation on held-out data (a train/validation/test split). If you are having questions about the proper model evaluation (which is usually the case when it comes to statistical significance testing), make sure to ask us. We are known to be very judgemental towards projects with poor evaluation.

The topic of each project has been addressed in the past by many researchers. Hence, you are required to check how your system fares against the best available systems out in the wild. Luckily for you, that usually means just using the reported scores from past competitions, as all the participants are evaluated in the same way. Additionally, it is a common practice to evaluate your system against terribly simple models, often referred to as *baselines*. This comparison serves as a sanity check: does your super-duper ultra-complex model actually perform better than the stupidest model does? Baselines often include dummy classifiers that return the majority or a random label, regression models that always predict the mean target value, and the like. We also welcome multiple models, whether the simpler (or simply different) versions of your main model, or something completely different.

Lastly, your project is always more interesting if you include different experiments apart from the official evaluation. For instance, you could carry out an error analysis in which you discuss why your model fails on some of the examples, or make an analysis on how different features or architecture tweaks affect your scores. Sky's the limit!

### 1.5 Writing Up the System Description (or a Research!) Paper

No matter how majestic your code may be or in how obscure programming language it is written in, your research will fall flat if you don't present it correctly. In other words, what makes your work recognizable is your scientific paper, or, in our case, your system description paper. Nobody will go through your code unless they are trying to re-implement your system. That means you should try your best to present your work in a perfectly understandable, succinct, and (possibly) self-contained manner. Whatever might seem ambiguous or dodgy, the reviewers (the teaching staff) are invited to assume the worst, i.e., you won't be given the benefit of the doubt. This means that, if you don't mention that you have done a certain thing, we won't know that you've done it, what will in turn lower our impression of the work done. Having a good title is also a part of making your paper recognizable: don't use generic titles like "Sentiment Analysis Using Support Vector Machines", but rather give it a unique flavor.

[3] http://scikit-learn.org/

[4] http://www.nltk.org/

[5] https://spacy.io/

Note that this isn't our "contraption" – insisting on elaborate descriptions is perfectly aligned with the practices of any conference or any journal: when you submit a paper, it gets reviewed, and you get your work accepted or rejected based solely on what you have written in your paper. So, please make sure to give your best to write a good-quality system description paper. Ideally, your information in your paper should be sufficient for any other researcher to reproduce your system and report the same performance.

Your paper will be what we call a "system description paper". What this means is that it focuses on describing the workings of a functional NLP/IR method for a concrete task, and does a fair job in evaluating it and comparing it to other approaches. This is contrast to a standard research paper, which presents a new method, a problem, or both. In other words, the crucial difference between your paper and a standard research paper is that your paper will most likely lack on the *novelty* aspect. This is fine. However, this doesn't mean that the paper is destined to be lackluster and useless. No paper should be useless. What makes a paper worth reading is the *contribution* it makes, and a contribution need not necessarily involve a new method or problem – rather it can be an insight of any kind, as long as it's potentially useful to others. What this ultimately boils down to is the *research question* (or questions) that the paper sets to answer. Thus, instead of simply describing a system that addresses a long-standing NLP/IR tasks using a well-known method, why not come up with a couple of interesting research questions that nobody has looked into before? For example, maybe you're doing sentiment analysis on tweets and use standard ML classifiers and standard features for that. Well, maybe no one looked into how emoticons interact with tweet topics, or into whether sentiment is more difficult to predict for male users than for female users, or for non-native speakers of English, or for teenagers, etc. Or maybe you're interested in implementing a bunch of models and compare how they perform. That's fine, but why not frame it as a more interesting research question? For example, you might contrast traditional, shallow ML models and more recent, deep learning models, and analyze which work better on what sort of tweets. Or maybe you can compare the effect of different word embedding vectors, so that other researchers learn what works and what doesn't without having to try it on their own. Sky is the limit here when it comes to inventing interesting and useful research questions. Another useful thing to think about when writing interesting and useful papers is to think about the *story* that the paper is telling. Humans love good stories, and scientific papers are no exception here. A good paper tells a good story. The story should engage the reader, in the sense that it makes clear why your research questions are worth the effort, what motivates them, and what we can learn from your results. If you manage to organize your paper around a couple of interesting research questions encapsulated in a good story, you will effectively transform a dull system description paper into a proper research paper, which is much more fun to read!

Deep thoughts aside, you'll be required to typeset your paper in LaTeX, using a template we'll provide you. Detailed instructions will be given in the template. There will be a limit on the length of your paper (exceeding this limit will result in **no points** for the paper). The paper is to be submitted via Easy Chair,[6] and will a part of the TAR 2021 *proceedings* (a booklet containing all student papers), published on-line.

[6] https://easychair.org/

### 1.6 Giving a Presentation

What goes hand in hand with the paper is its presentation. The presentation should succinctly explain the task at hand and how you tackled it. For obvious reasons, we recommend trying out your presentation before the official slot and make sure you're able to get the message across. To spice things up, we ask for a mandatory system demonstration, as we are eager to see your systems *in action*.

## 2 Milestones

To make sure you're working on your projects continuously (and not cramming the 11 weeks of work in the last two), we introduced a few milestones. These milestones were not introduced on a whim – it's not like we adore increasing our already ever-increasing workload. They are here solely for your benefit!

### 2.1 Project Checkpoint Alpha

April 4, April 5

This project checkpoint is as a **mandatory** preliminary sanity check of your progress. For the checkpoint, you are required to have an initial plan of what you intend to try and have a portion of preliminary preparation work done (i.e. familiarize yourself with the data and related work).

### 2.2 Project Checkpoint Beta

May 23, May 24

The project checkpoint stands as a **mandatory** final check before the final project submissions. For the checkpoint, you are required to already have a certain portion of work done: you should *at least* have a working baseline system (or a couple of them in case of multiple sub tasks), evaluated using the official evaluation metrics. The checkpoint basically amounts to you presenting us in short what you've done so far and how you plan on proceeding. We'll also give you the final instructions on how to pimp up your project. It's important to mention that we require that all the team members show up at the project checkpoint; let us know if that's not possible.

To enforce that you work continuously, if you don't complete the things mentioned above, we'll deduct 25% of points from your final score. Of course, you could probably get away with some ridiculously simple baselines, but that's not the point of this project (or education in general). Please, do your project industriously and create something you'll be proud of later.

## 2.3 Project submission

**Deadline: June 5**

You will upload your paper in PDF format and (possibly) accompanying LaTeX code to the designated place. Instructions for this will be posted on FER Web a few days prior to the deadline.

## 2.4 Project Presentations

June 6, June 7

You are given 10 minutes to present your work in a succinct and interesting way. You should shortly introduce the task, your approach in tackling it and the achieved performance compared to other available systems. Please make sure to make the presentation smooth. It's up to you which team member will present your work. As for the language, we encourage you to leave your warm and cuddly comfort zone and present your work in English. We won't force you, but we think it's the way to go. (In our honest opinion, it's far better to "embarrass" yourself in front of the teaching staff than in front of your potential employer.) Additionally, it is *mandatory* to attend all presentations, be it the one of your team or that of others. We think it would be rather stupid for the team to present their work only to us and not see what the others worked hard on for over a hundred hours. ☺

Project presentations *must* include short demonstrations (at most five minutes). We don't expect anything fancy, but we do expect an interactive demonstration that is interesting both for us and for the audience (feel free to include witty examples). This may include almost everything: from a relatively simple interactive command-line application to a tiny web-application. It's up to you! Of course, you should make it interesting as lazy or boring demonstrations can hurt your presentation score. For instance, running model training and evaluation code in a Jupiter notebook is not interesting; preparing peculiar examples and coming up with new ones on spot is. In short: we love interesting interactive demonstrations.

## 2.5 Project Reviews

Your system description paper will be evaluated by three reviewers, on of which will be a student teaching assistant. We'll aggregate reviews and send them to you via Easy Chair.

The reviewers will tell you about the things you *must* address, as well as those that would be nice to address to make your paper stronger. This will mostly cover the content, but it might also cover the language and typography. We enforce typography and style fixes to ensure a uniform style of our proceedings (and we really, really, really love the proceedings to be impeccable ☺). If you don't agree with some of the reviewers' comments, please let us know as soon as possible.

## 2.6 Final Papers

You're required to address the remarks brought up in the reviews and produce a final (so-called *camera-ready*) version of your system description paper. This might take a few iterations and will not affect the score you already received from the reviewers, but it's **a prerequisite for passing the course**.

## 3 Evaluation

The final project score is obtained as a weighted sum of four components: technical soundness (35% of the grade), substance (35%), paper quality (20%), and presentation (10%). Again, note that we evaluate technical soundness and substance based solely on your paper (not on your code or presentation). After all, the reviewers of your paper won't check your code and won't necessarily attend your presentation. Hence, please make all of these perfectly aligned in content. The final score is calculated as follows:[7]

$$\begin{aligned}\textbf{score} = ((0.35/5) \times \text{technical soundness } + (0.35/5) \times \min(5, \text{substance}) +\\ (0.20/5) \times \text{paper quality } + (0.10/10) \times \text{presentation}) \times 55 + \text{bonus points.}\end{aligned}$$

Maximum number of points is **55**, with bonus points given (on top of regular ones) to the teams that put more effort in their projects than required (up to **2** points). Score components that cover the work performed (the first three ones) are explained below.

**Technical soundness (35%).** This component assesses the correctness of the approach used to solve the project task. Mistakes penalized here are, for example, not calculating the inter-annotator agreement (AA) where needed, using equations that are just plainly wrong, not optimizing the hyper-parameters, optimizing them on a test set, testing on a train set, and not performing a statistical significance testing. The possible values are:

- 5 = Excellent and professional. Everything was done by the book, from the model, features, and model selection, all the way to the evaluation.
- 4 = Very good. Most of the work was done properly, but there are couple of details that could be improved upon.
- 3 = Good. The sole idea is okay, but significant faults can be found in the execution.
- 2 = Enough for the passing grade. The underlying idea is somewhat okay, but there are some grave mistakes in how it was carried out.
- 1 = Not enough. A team of (somewhat) trained pandas would have done a better job (seriously).

**Substance (35%).** This component is concerned with the amount of work a team has done. Not doing everything that is asked in the project description (e.g., testing one model instead of two) is penalized. However, doing the bare minimum is also not welcomed (e.g., taking the simplest of models just to meet the requirement). Make sure to give your absolute best. Also, what we value here is the effort you make to shift from a bare system description paper to a proper research paper (cf. subsection 1.5): an interesting story that revolves around useful research question is what adds substance to your work. This component takes into account the team size (two or three students) as well. The possible values are:

- 7 = The extra light-year. It's like an extra mile, but even more extra. What started as a lowly TAR project, became a comprehensive and thought-out project going well beyond what was initially required. We salute your efforts!

[7] Substance score is capped to 5 when calculating the total score, but having 6 or 7 in this category earns 1 or 2 bonus points, respectively.

- 6 = The extra mile. Not only did the team do everything that was asked in the project description, but they also included some additional models or experiments at their own initiative. We think that this is definitely worthy of an extra pat on the back.
- 5 = Excellent. A solid amount of work was performed and everything from the project description was completed in adequate detail.
- 4 = Very good. A nice amount of work was performed. However, there are either some minor things missing or the work seems to be done in a really lazy and unimpressive manner.
- 3 = Good. A non-trivial amount of work was performed, but there are still some significant portions of the project missing.
- 2 = Enough for the passing grade. Barely enough work was put into the project and there are considerable portions of it missing.
- 1 = Not enough. Even a sloth dozing in a tree fork would put more effort into this.

**Paper quality (20%).** This component assesses the quality of the system description paper, accentuating how the work was presented and explained rather than the work itself. Mistakes penalized are, for example, unclear explanations, missing technical details important for reproducible of the work done, senseless references, typos, and false statements. When evaluating this component, we will of course take into account that this will be the first paper ever for the majority of students. The possible values are:

- 5 = Excellent. Everything is both peachy and hunky dory.
- 4 = Very good. A well-written paper, up to a few pedantic details.
- 3 = Good. An okay paper holding a lot of room for improvement.
- 2 = Enough for the passing grade. A badly-written paper that barely manages to get the message across.
- 1 = Not enough. The paper might as well be in Hungarian, as it wouldn't make any difference in clarity.

## 4 Dissemination

We plan on publishing all of your system description papers in the course proceedings. These will be indexed by many search engines, Google Scholar included! Considering there's quite a few people with an internet connection out there, make sure to give your best when working on your projects and papers. We recommend writing the project report in English, as it will make you more confident in your language skills and give you greater exposure. However, we cannot publish your work without your consent, so please let us know as soon as possible if you are against it. If we don't hear from you, we'll consider that you are okay with us publishing your work.

Additionally, we are aware of the fact that some of you curate your own Git Hub profiles and that you would like to make your work publicly available. Generally, we are totally okay with that unless we've explicitly provided you with some data that should not be publicly disclosed. (For instance, according to the Twitter Terms of Use, tweets may never be distributed, as their authors should be able to delete them for good.) In that case, as long as you don't publish the data along with your code, you're good!

## 5 Important Dates

- March 4: Topics announced
- March 4–13: Bidding for topics (in teams)
- March 15: Topics assigned
- Apr 4, Apr 5: Project progress checkpoint alpha
- May 23, May 24: Project progress checkpoint beta
- June 5: Project submission deadline
- June 6, June 7: Project presentations