**General API Problems:**

Most of the metadata services for retrieving academic releases were either paid, were free for some time or with some limits and then paid, or did not have an API at all (Google Scholar). The APIs available mostly did not have tags about the releases. They sometimes included abstracts or short descriptions, but not always. In order to retrieve the releases that we wanted in a reliable way, we had to come up with a way to choose releases only based on their titles, since this was the only information that seemed to be present for every single release. Here we encounter a fundamental problem.

Let us consider the following example. We pick only “deep learning” as a topic to search for in the releases. If we simply try to retrieve releases, whose titles contain “deep” or “learning” or “deep learning”, we would miss a significant amount of releases which are indeed about deep learning. For example the paper “Attention is All You Need”, which is a revolutionary paper introducing the transformer architecture and attention blocks, would be skipped. Its title neither contains “deep” nor “learning” nor “deep learning”. By skipping this release, we would also potentially skip all the exceptional people who contributed to it.

As this example demonstrated, it is clear that we have to find a different way than simply checking if parts of a topic (or all of it) is in a release’s title to retrieve it or not. To achieve a better way of retrieving relevant topics we came up with the sub-concepts solution.

**The Sub-concepts solution:**

The general concept is to generate sub-concepts about a concept, which could come up in the title of a release. This is easier said than done. In order to achieve this we tried two approaches:

**1) Wikipedia:**

This first approach was meant to be more of a placeholder in the early stages of development. The idea here is getting links from the Wikipedia page of a concept (if it hopefully exists) and using these links as sub-concepts. Obviously, this is a terrible way of doing this, because the links are not filtered in any way, so we have a lot of random unrelated words like “Xbox”.

**2) ChatGPT:**

After careful consideration, we decided to use the ChatGPT API to generate sub-concepts related to a concept. With the right prompts, we managed to get quite reasonable sub-concepts about various concepts. We will not be going into detail about how reasonable these results are, since determining their quality is out of scope for this paper.

**Choices of API:**

**Fatcat API:**

Out of the APIs that we had researched, initially we have decided on working with the Fatcat API, because it was completely free and seemed to contain the most amount of academic releases from various different sources. Sadly, we had many problems specifically with the Fatcat API, which we will discuss now.

**Uniqueness problem:**

It is known, that there are many people in the world with the same name, and even people with the exact same name and surname. Therefore, an attribute unique to every person is necessary, in order to be able to distinguish different people. When we think about our specific topic, which is looking for qualified researchers about various topics, ORCID comes to mind. ORCID stands for Open Researcher and Contributor ID, and an ORCID iD is “a unique, persistent identifier, free of charge to researchers” [citation to ORCID about page]. Many researchers have an ORCID iD and this could be used to identify them.

When we get a release from Fatcat API, we also get a list of contributors to that release. The problem is the fact that we usually only get the names and surnames of the contributors and not a unique identifier like an ORCID iD. Therefore in such a case, distinguishing researchers who share their names and surnames is outright impossible. Also, the names and surnames would very often be abbreviated, and not in a common way. Sometimes the name would be abbreviated, sometimes the surname, sometimes both of them and sometimes none of them. Because of this problem, we initially tried the following workflow:

1. Merge researcher entries who have the same name and or same surname, which we have attempted to parse from the given data from Fatcat API.
2. Mark every merged researcher entry in the resulting database with a flag such as: “multiple\_authors”: True.
3. For all researcher entries that are marked with this flag, look up their names using the ORCID API and save a fixed number of ORCID iDs to the entry, so that a user can at least try these iDs to find the author that they are actually interested in.

A problem with this method is, because we do not even get complete full names and surnames from the Fatcat API, we would merge much more entries than we would need and we would have no way of avoiding this. Another problem is, since again we do not have the full names and surnames, the results that we get from the ORCID API seemed to be not so reliable and it seemed that there was no guarantee that the first results were the more “relevant” ones. Also, using the ORCID API for every single merged researcher entry also affects the performance considerably, which is in our case speed.

**Speed problem:**

Another problem that we encountered with the Fatcat API was an expected one. Since we are usually asking for millions of releases, it simply takes long to receive them all. We could manage to get a maximum of 10.000 releases per response from the API and receiving a response would take a couple of seconds, let us assume 20 seconds for now. So as an example, if we wanted to receive about 10.000.000 releases, this would take (10.000.000 / 10.000) \* 20 = 20000 seconds, which is more than 5 hours.

Because of the aforementioned many, many problems, we have decided to use a different way to access authors and releases, which would hopefully solve most of these problems.

**ORCID Public Data File:**

**TODO write down the structures of the activities and summaries files.**

**TODO display some meta-information about the files.**

**TODO diagram showing how the whole process works.**

ORCID makes all its data public every year around the end of the year. This data includes the records of all the authors with their respective ORCID iDs, their included releases and more. The amount of releases and authors that we will be able to get from this source seemed to be less than with Fatcat API, but we have made the assumption that the people we are looking for probably have ORCID iDs and upload their releases there fairly frequently and so the difference was not so important. By using ORCID records, we could be certain that we would have no duplicate researchers in our entries (unless someone has duplicate ORCID iDs, then we cannot do anything) and we could connect them to their respective releases much easier. There would also be no way of having duplicate releases, since ORCID uses a field called “put\_code” in its data file, which is unique to every single release. In order to use the ORCID Public Data File, we first had to download it, extract it and save the relevant information in a local mongodb database. [MODIFY THIS] This was a relatively challenging task, since there were files which were too big to extract locally, so we made use of the “tumcslecture” server to perform the extraction. [MODIFY THIS] So with using the ORCID Public Data File, our workflow was as follows:

1. Save all ORCID data in a local mongodb database.
2. After generating sub-concepts from given concepts, search for ORCID releases in the database, whose titles include a subconcept.
3. After finding the relevant releases, search for authors, who are the owners of these releases.
4. Continue with the second step.