Zeta Avarikioti*, Roman Brunner, Aggelos Kiayias, Roger Wattenhofer, and Dionysis Zindros

The iceberg below the surface: An analysis of dark net content

https://v2.overleaf.com/project/5b6c19affe683e5bba4ed91b

Abstract: We analyze the type of content present on the "dark web", the set of websites accessible via Tor. We create a darknet spider and crawl the darknet starting from a bootstrap list and recursively following links. The whole connected component of more than X websites and Y base URLs is explored. We publish our spider as open source software. We find that the darknet is well-connected through hub websites such as wikis and forums. We perform comprehensive measurements on the content found using machine learning to analyze and categorize the various types of content. We observe that the majority of darknet content belongs to P and Q. [1]

Keywords: anonymity, tor, machine learning, scraping, spider

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

When talking about the darknet, one does often think of the Tor network in particular. However, there are other so-called darknets provided by other anonymisation networks such as i2p[2] or FreeNet[3]. So what is the darknet and how is it defined? Let us first have a look at the whole iceberg and start with the tip of the iceberg above the surface. This part of the internet is typically referred to as the clearnet, which may be best defined as the pages that one can find directly or indirectly using their search machine of choice. The part below the surface is often called the "deep web", which contains everything that is not directly accessible without the further use of tools or knowledge. This means that only a restricted number of people can access those pages. Such areas could be a university's internal network (may require tools (VPN) or passwords) or a personal eMail (may require a password or card reader - and is most of the time not public). Although this is arguably a large part of the deep web, there is also another part of the deep web which requires additional software, such as the Tor browser, to be accessed with the intention of preserving the anonymity of the users. This part is often called the "darknet".

In this work, we look exclusively at the Tor network since both the public and the academic community has a high interest in knowing what the Tor network is used for and there is still a lot of speculation going on about the content as well as the structure of provided pages on the Tor network. Furthermore, it is also the probably best-known darknet network and hence is often exclusively referred to as the darknet.. The Tor network can be used to access any page on the internet (that is publicly available), so it is often used to access pages from the clearnet, while remaining anonymous [4]. However, Tor also offers a mechanism to provide services anonymised. These services are called hidden services (HS). In order to access an HS one has to know its exact URI, which either consists of the hash of the RSA public key of the webpage (as the hostname) (v2 address) or of the base32 representation of ed25519 public key concatenated with a checksum. Both versions have .onion as their top level domain, even if it is not a registered TLD, it is still reserved for the Tor project. This makes it very hard to guess hostnames and almost impossible to test all in order to brute force access hidden services. However, some of the pages actually want to be found. Therefore we can partition the network into two subgraphs - the visible part, where pages advertise its URI on link hubs and search engines and the invisible part, which contains only pages with URIs that are not publicly listed.

We focus on the visible part of the Tor network in this work, since even though there exist techniques to collect hosts from the inivisible part of the darknet by running a larger number of guard nodes [reference OWEN], we consider the information of such HS to be

1 [[maybe add another sentence on the results of content classification when we have them.]]

***Corresponding Author: Zeta Avarikioti:** Affil, E-mail: zetavar@ethz.ch
Roman Brunner: Affil, E-mail: robrunne@student.ethz.ch
Aggelos Kiayias: Affil, E-mail: aggelos.kiayias@ed.ac.uk
Roger Wattenhofer: Affil, E-mail: wattenhofer@ethz.ch
Dionysis Zindros: Affil, E-mail: dionyziz@gmail.com

2 https://geti2p.net/en/

4 Anonymity also includes resistance to censorship of other behavioural and technical changes. Note that the level of anonymity always depends on the anonymity set.

private and therefore should not be accessed by anyone without the consent of the owner. In addition to that, we want to classify the part of the Tor network that everybody can access without additional information. That means that we do not want to classify content from invisible HS since we might find and classify contents from email inboxes or address lists, which is not of public interest and often requires further authentication via password of HS authentication (knowledge of the content of a cookie). If we only look at the visible part, we only find data that is publicly available. We assume that the visible part of the darknet has a similar link structure to the clearnet, where pages link to each other. Therefore, we describe how a scraper can be used on the visible darknet, which components are necessary and which information can or cannot be found. A scraper requires fewever ressources than previously suggested collection techniques [OWEN again] and reduces the impact on the network since it only acts as a regular Tor client.

We compare our results to previous results, which exploits attacks on the Tor network or guard nodes for data collection. This comparison serves as an indicator of whether a scraper works on the darknet and as a proxy to check if the collected dataset is representative. Furthermore, we estimate the size of the darknet (number of HS) based on the visible HS found and compare it to the estimates provided by the Tor project[5]. Previous work in this field operates a large number of guard nodes, which knows the actual address of an HS [6][7]. In contrast to our approach, this method collects any found HS and cannot fully reconstruct the link structure of the Tor network since it only sees its share of the whole network. However, it allows to test how often particular pages are accessed, a measurement we cannot provide. Other work in this direction that targets darknet content classification uses a flaw in the Tor protocol in order to collect data[8]. This flaw was fixed in 0.2.4.10-alpha. Therefore this entry point for data collection is no longer available. Furthermore, exploiting a flaw in the Tor protocol contradicts our goal of collecting data with the least invasive method for the Tor network.

The scraper is newly developed for this work and made public under MIT license for further use by other researchers[9]. It implements a complete data analysis chain, from collection over data preprocessing to classification. The data collected not only contains the content of the HS but also the link structure, status information, network topology information and can be configured to store the change over time of the network.

2 Scraping the Darknet

In this section, we describe in detail the structure of the spider we built to crawl the darknet. To understand some of our design choices, we present the problems faced and the different strategies applied to resolve those issues. Furthermore, we present some preliminary results on the darknet structure we collected from the crawling process.

The goal of the scraper is to download content from as many hosts as possible. In order to discover more hosts, we download not only one page per host but multiple, since those can contain further URIs leading to the discovery of new hosts. The goal is to build up a dataset consisting of contents and the network structure itself. We therefor do not only store the downloaded content, but also the links between the different pages.

We seeded our scraper by inserting a list of 20'000 host addresses from previous work[10], which allowed us to reach a high scrape speed right from the beginning. Out of those 20'000 hosts, approximately 1500 were still online.

To get a complete scrape of the visible darknet, we needed at least the following functionalities:

- Network module: The functionality of this module is to download any content and handle any network related exceptions and issues.
- URI extractor: It is used to parse the downloaded content and extract URIs that will be used recursively to download further content.
- Database: Stores content downloaded as well as the network's structure and URIs to download.
- Tor proxy: Since the scraper should download hidden services (HS), a Tor proxy is required.

5 Statistics released by the Tor project concerning the number of available HS: metrics.torproject.org/hidserv-dir-onions-seen.html

6 https://www.torproject.org/docs/onion-services.html.en

7 [[reference]]

8 [[reference]]

9 Git repository

10 [[insert source here]]

The modules depicted in Fig [] provide this functionality and are explained below in more detail.

2.1 Downloading the content

The network module needs to connect to the Tor network, therefore needs a Tor proxy. Since the Tor network is slower than a typical network connection[11], we used multiple Tor instances, to increase the download performance of the scraper. The Tor proxy spins up a configurable number of Tor instances and uses a round-robin scheduler to distribute download tasks across the instances. When testing, we were testing with only a small set of sample pages. Since a page often links to itself, this lead to a state, where we issued a lot of concurrent connections to the same host. In order to prevent such DoS like behaviour of our scraper, the network module has to keep track of how many connections are made to a specific host in order to prevent such behaviour. If it detects that there are already four requests pending to a given host, it puts the new download task into a waiting queue, which will be consumed if a connection to this host is freed up. This works good, if all pages are of similar size, but we ran into a condition, where we started loading large parts of the database into our pending queues. Therefore, the other modules have to ensure the network module is not overloaded with too many requests to the same host, since the memory usage of the network module increases with every request in the waiting queue. The value of four concurrent requests was chosen to be lower than the default Tor Browsers setting[12] to be less detectable by scraping countermeasures some hosts apply.

Since we download any URI we find on the darknet, we do not know anything about the legality of the content, before we have downloaded it. In order to prevent storing illegal data such as child pornography or movies protected by intellectual property rights, we have a set of guidelines that the network module has to follow. First of all we only want to store textual content, since this minimises the risk of storing illegal image, video or audio data. In order to achieve this, the network module has the following filters in place to prevent the download or even worse the storage of illegal content. First, the network module will check the returned content type header. If one is supplied and it is not a textual representation, the download will be immediately aborted and the buffers are flushed immediately. If no content type header is specified, we try to guess the mime type of the downloaded data as soon as it is available. if the data is again not of textual representation, it will be immediately discarded. So far we only downloaded text files, may it be an html file or a txt file. However, one possibility exists, how we could still store media content and that is by data URIs such as base64 and similar. Therefor we make an additional pass through every string downloaded and replace any data URI that does not encode text with a placeholder. These filters are here to ensure now illegal content is stored by our database.

2.2 Orchestrating the scraping

The conductor balances the needs of the other modules. It is the responsibility of the conductor to dispatch URIs for download to the network module and get new URIs from the DB. Since the network is typically slower than local storage (and this is especially true for the Tor network[13]), the goal of the conductor is to ensure that the network (as the scarce resource) never has to wait for local storage to deliver new URIs for download. To balance out short peaks in access time to the local storage, we use a pool of URIs ready to download. The size of the pool is configurable to adapt to different setups, e.g. an SSD can deliver data much faster than an HDD; therefore the pool size can be smaller, which also results in a smaller memory footprint. We noticed early in this work that deciding which URI to schedule next for download is important. We also observed that at some point during the scraping process most of the available URIs only stem from a few hosts. Since we do not want to DoS any hosts and still be able to scrape fast, we require a way to prioritise URIs. The prioritisation scheme should ensure fast discovery of new hosts and paths while it avoids bombarding single hosts with tons of requests. These requirements lead to the definition of the following prioritisation schemes:

– **Random**: When scheduling the URIs in insertion order, more than the allowed number of URIs are dispatched for download almost immediately, since a page often contains many links to itself. One easy and fast way to mitigate this issue, and therefore our

11 https://metrics.torproject.org/torperf.html

12 Tor 7.5.6, setting network.http.max-persistent-connections-per-server

13 Statistics released by the Tor project concerning download speeds: metrics.torproject.org/torperf.html

first approach, is randomising the order in which the URIs are scheduled. However, this strategy breaks down when large amounts of the available URIs in the DB are pointing to only a few hosts. In such a scenario, randomised access fails, since even if the URIs are reshuffled, they still stem from only a small group of hosts.

– **New**: We also discovered, that the "Random" strategy stalls pretty fast by means of reaching new hosts. The main reason for this are hosts we call "black holes" that have a tremendous amount of links to themselves. An example of such a page is a bitcoin explorer, which lists action ever recorded in the bitoin's blockchain. So we want a strategy that does not go deep into a single hosts pages, but more into the depth of the network between the hosts. In this strategy, we only get paths from hosts that are not yet scraped, ie we did not visit earlier any of the available paths from this host. Therefore we only download one page per host and do not dispatch another path of the same host afterwards. This method is efficient when the collection of not yet scraped hosts is huge because it advances very fast through the network. However, the first page downloaded from a host is not always useful for classification or collection of further links. Therefore this strategy only works well for seed generation but not for data collection.

– **Prioritised**: Since the "New" strategy does not guarantee to generate good data for link collection or classification, we attempted to use another part of the dataset we collected along the scraping progress to serve as an indicator of which path should be dispatched next. Along each URI, two counters are stored. One counts the total number of links pointing to this URI, while the other only counts links from distinct hosts that point to this URI. These counters then are used as a proxy for the importance of a particular page. We consider a high incoming link count from different hosts to be a high vote that this page is more important then others. Although in general this strategy is efficient and results in a power law distribution, which would give us a nice prioritisation schema [– see powerLawAbnormality.png], we observe a few abnormalities. The most obvious abnormality stems from bitcoin scamming pages linking to a bitcoin explorer, in an attempt to prove that their credibility. Such tightly connected hosts can break this prioritisation scheme, by introducing plateaus in the distribution, therefor the scraper will be busy downloading contents that will link again to the same pages.

– **Recursive**: In an attempt to make equally progress on all hosts, we introduce the "Recursive" strategy. This is one of the most advanced strategies, and thus one of the most computationally and IO intensive, but still linear in the number of input hosts. In this strategy, we select one path per host, if the host has not yet a request pending and still has unscraped paths available. As soon as the pool needs to be repopulated, we go again through all the hosts and check, which ones are ready for a download. The idea is to dig deeper into the single hosts level by level. This strategy passes through the single hosts as if they were a linked list and ensures that we make progress on all hosts in a similar speed. New hosts are considered as soon as the pool needs to be repopulated. Since a webpage of any given host looks more like a tree to our scraper (we do not follow links back to pages we already visited, we only note that this link exists), we have to decide which of the children of a host to consider next. We decided again to look at the incoming link count of a path. The recursive strategy always chooses the path of a host which was not yet scraped and has the highest incoming link count from distinct hosts. The idea behind the link count as a proxy is similar to the one in the "Prioritised" schema.

– **Inverse recursive**: This strategy is the same as the recursive strategy, except it chooses the path with the lowest incoming link count per host. One can choose this option to try and make faster progress on the network scraping since a path that has a high incoming link count is already part of a well-scraped area, whereas the one with the lowest incoming link count is probably part of a not yet very well discovered area.

– **Combined**: Since both recursive strategies can take some time, especially on slower hardware, we came up with an approximation to the "Recursive" strategy. This strategy combines the aforementioned "New", "Prioritised" and "Randomised" strategy to mimic the recursive strategy with less computational overhead. First, we apply the new strategy until we are not able to repopulate the pool anymore with only not yet scraped hosts. Then, the prioritised strategy is used. If this strategy gets stuck (e.g. because it hits a region of interlinking hosts), it will not be able to return enough paths to repopulate the pool. In this case, this strategy falls back to the random strategy, getting entries at random.

The proposed combined strategy is an approximation to the recursive strategy; we use the new strategy first and we provide multiple starting points for the prioritised strategy to ensure we do not immediately run into the issue of lots of interlinking hosts. However this issue can still occur, thus the fallback to the random strategy is important to keep the scraper running. Note that the lower strategies are always applied first, so as soon as a new host becomes available for download, it will be dispatched. Although, this is not a perfect replication of the recursive strategy, it is less computational and IO expensive; however, it is not as precise as the recursive strategy and in the worst case fails with the issues the "Random" strategy has.

2.3 Extract URI

In order to keep the scraper running and getting new targets to be downloaded, we need to extract URIs of the downloaded pages. First we applied regular expressions to match HS URIs in a string. However, such HS URIs can get quite complex which also introduces a high complexity to the RegEx needed. We found that especially if the downloaded text is very large, it can take quite some time to extract the URIs using such a RegEx, up to several seconds. Since we need to extract at least some portion of the URIs fast enough to introduce new URIs in at least the speed data is downloaded, we needed a faster way to extract data running along the scraper. We then measured how long it takes to extract links (that are correctly tagged in an html) with cheerio[14]. To ensure we only collect links of HS, we have a smaller RegEx ensuring that it is not a link to the clearnet. This approach was ten to a hundred times faster than the RegEx based implementation, however it had the shortcomming, that it did only capture links that are part of an <a> tag in the html string. Since some pages return some other textual representation or do not correctly link to other pages (e.g. not enclosed in an <a> tag), we added another component, running the slower RegEx based technique, capturing every link of a page, in parallel to the scraper. This enables us to keep the scraper running at all times by feeding it with new URIs from the cheerio based scraper and still have the complete extraction of the RegEx based extractor. Note that the time it takes to extract the URIs using the RegEx approach typically takes about the same time as the download of this specific content, however, there are pathological examples, where it takes way longer to extract the URIs. Therefor, we needed the faster extracting cheerio based implementation, since this implementation has a better predictable timing behaviour.

2.4 Storing darknet content

For storage we used PostgreSQL, a relational database system. The powerful query language provided helps a lot in integrating different aspects of the system such as the prioritization schemes. In order to store all downloaded content as well as the link structure, we introduced several tables. First of all we have the baseUrls table, which stores all the host addresses and a denormalized count of the number of hits, which describes how often this host was referenced. In order to specify a full URI we now need the path, therefore we have the paths table, which is the main entry keeping the state of an entry. In this role, it contains a flag indicating if a download for this specific path is in progress, a timestamp indicating when the last successful download finished, an integer indicating at which depth of the graph pass that the scraper does on the darknet the entry was found, the path itself as well as a possible subdomain. Additionally, it contains the two denormalized counters for number of hits and number of distinct hits used by the "Prioritised" priorisation schema. In order to store the downloaded content, we have the contents table, which stores the content itself as well as some metainformation about the content such as the statusCode of the download and the content type. In order to allow for multiple downloads of the same path, if one wants to track changes of the network over time, it has a scrape timestamp. Lastly, we need a way to store the topology of the network, for which we have the link table. The link table establishes a n-m mapping between the paths table, since one page may contain multiple links to other pages which can be uniquely identified by their path and the base url.

2.5 Accumulated data

Using the above described scraper we were able to find 34'714 host addresse, of which around 10'000 (multiple runs resulted in between 9'100-10'900 responses) hosts responded to the http(s) request. Out of those hosts, 7'200-7'600 responded with content that was actually

14 https://cheerio.js.org/

useable (i.e. not containing an http error status code or an empty html page). We collected a total of 7.8×10^6 paths and 71.35×10^6 links. Of those we downloaded 1.5×10^6 contents. The remaining not downloaded paths all belonged to hosts that we classified as black holes hence are unlikely to contain further URIs to unknown hosts.

Type of data	Number of instances in dataset
Host addressed total found	$24'714$
Hosts responding	$9'127 - 10'957$
Hosts returning useable content	$7'299 - 7'566$
Paths	7.8×10^6
Links	71.34×10^6
Contents (Downloaded paths)	1.6×10^6
Content (useable)	$626 \times 10^3 - 667 \times 10^3$

We find that the visible part of the darknet is actually well connected. There exist pages that we call hubs that have much higher outgoing link counts than other pages. Such pages are often link lists similar to the hidden wiki, which play a central role in the navigation of the darknet. Such pages also link to a large part of the visible darknet directly. The page listing links to the most other hosts responding contained 55.8% of the online hosts we found. However, such pages always have a large number of not working links, since once a URI is added it is rarely ever removed again. Despite the fact that more than half of the darknet can be found in just one step if one selects the right entry point, we saw a maximal depth of 22 until we found the last host in our dataset. Graph i shows how the hosts are distributed over the depth. Note that this does not mean that this is tha maximal distance between two nodes, since there possibly exists a faster way from depth 0 to depth 22 than the one the scraper has taken.

3 Analsys methodology

Given the size of the collected dataset of 667×10^3 contents, we are not able to classify such an amount of data manually in a reasonable amount of time. Therefor we decided we need to employ machine learning techniques to achieve the classification of a large portion of the dataset. In order to be able to classify the pages based on its textual contents, we needed to preprocess the contents first.

In order to complete data analysis, we neeeded at leas the following parts:

- A data preprocessor to normalize the downloaded content
- A classifier that classifies the contents based on the preprocessed intermediate representations
- A database to hold both, the intermediate representation as well as the classified results

The modules depicted in Fig [] provide the above functionality and are explained below in more detail.

3.1 Preprocessing darknet web content

The preprocessor has two main tasks to accomplish, first it has to linguistically preprocess the contents and then to index the contents, such that they are ready for classification. The first step is extract the actual text from the content. This is done by stripping any HTML tags and other script tags we find in the input string, using cheerio. After getting a clean string, we need to find out which language the content is written in, in order to decide which preprocessing will be applied to the string. We use franc[15] and we use the minimal set of 82 languages. There exist larger language sets for franc, however they usually make the result of the language classification worse, since we have quite a large amount of short content strings, in which franc might decide to classify one string as scottish englisch and the other as british english and so on. Since we are not interested in which dialect a content is written but mainly the general overlaying language (e.g. english for the previous example), we decided to reduce the languages supported. Overlooking the whole dataset, we found contents in 63 different languages, however, about $\frac{2}{3}$ of the contents are written in english, followed by about 10% russian. For the complete list see Fig [].

For the following steps we need an array of terms instead of a string, therefor we now tokenize the content string. We do not apply complex heuristics here but simply split at ever space and some usual splitting characters such as /, :, . or @. Since we have no precise information about the topic of the string,it is hard to apply more sophisticated tokenization at this point in the process.

Now that the language of the content is known, we remove stop words according to the language. As we found out, some of the contents only contain stopwords, therefor they are empty after stop words removal. These

15 https://github.com/wooorm/franc

will be marked as EMPTY right away. After stop word removal we do normalize the contents further and apply stemming where appropriate. Since ... [16] suggests that stemming works not equally well on all languages, we only apply stemming for english contents.

Everything is now ready to start building our positional index. We decided to build a positional index, since the information is available and even if we don't use it yet it is cheap in terms of storage and computation. Further we would have to store the term frequency anyway, since we want to apply a bag of words model for the classification step, which can be calculated directly from the positional index. To make manual classification easier, we also keep the extracted clean content string, such that it can be directly presented to a human classifier, in order to get the labelled training dataset.

3.2 Storing indexes for classification

We stored the extracted contents in the new table cleanContents with a foreign key to the original content. The new cleanContents table has a ont-to-one assignment for both, the language as well as the primary class label. Since the decision if a document is legal is a binary decision, we directly used a boolean column to store the legal value (true corresponds to legal, false to illegal).

The goal in the trainings phase is to employ active learning, therefore we need to have an idea how sure the classifier is about a decision it made. Thus a column for both the legal and the label are inserted in which the probability output of the classifier is stored. This allows us to apply active learning techniques in the classification phase.

The positional index itself is constructed with four different tables. The first table is the list of terms, which has a one-to-many mapping to the postings. Each posting in the posting table has a foreign relation to a clean content and a term, therefor creating the tuple (term, doc id). In order to be a positional index, every posting belongs to many positions, describing the positions at which the reference term appears in the specified clean content (document).

With the model described above it is possible to apply machine learning either to a set-of-words, a bag-of-words or a list-of-words, depending on the needs.

16 [[reference]]

3.3 Classify darknet content

Describe the types of content that we found. Talk about our methodology in how we analyzed the content using machine learning and NLP. How were the categories chosen? Discuss how we rank base URLs as well as individual paths. Show diagrams (a pie chart or bars?) with the popularity of various content categories.

4 Darknet contents

5 Political and Ethical Discussion

Discuss the political implications of tor. Is it mostly illegal content? Does it matter? Discuss the ethical considerations of our research and make sure you mention that we didn't subvert the protocol itself but only collected public data.

6 Related Work

7 Conclusion

Summarize our results and discuss directions for future work. One issue with the proposed method is the high volatility of the network. We found that up to 30% of the found hosts switch from being online to offline and vice versa in only a week. This circumstance could be faced by adding a temporal resolution to the data gathering process. The provided code already supports temporal resolutions for the hosts and paths, however, one has to match the links by timestamp.One addition could be that along the existing data collection, one also stores to which round of rescraping any entry (whether it now be a host, a path or a link) belongs.

FORMATTING To speed up the scraping process, we ran 100 concurrent requests to the Tor network. In order to prevent overload on the Tor nodes within the circuit, we used a pool of Tor of equal size, such that per request one Tor instance was used. The Tor instances were scheduled in a round robin manner

Describe the architectural decisions put in designing the spider. Mention the technologies we're using (node.js, postgres, the tor libraries). Include a diagram for our software architecture. Discuss how we bypass rate limits (e.g. by randomization of visits and by using multiple tor circuits in parallel) and what the rate lim-

iting problems are (including rate limits by individual websites as well as by the tor circuit / network itself). How we avoid downloading illegal content (whitelists and blacklists, content types). Talk about bootstrapping the list, exploration depth. Show the numbers such as how many sites exist, how many base URLs exist, how many links exist, summarize them nicely in a table. Try to aggregate data from our database to create insights for the reader. Extract the "hubs" and state that, contrary to popular opinion, the darknet is well connected; specifically discuss which this hubs are, what their purpose is, and how many links they have for the top 10 hubs.

8 Editorial Policy

8.1 Choice of reviewers

The Editor responsible for a given area of physics, turns to experts of the subject for opinion. Research articles and communications are reviewed by minimum two reviewers, review papers by at least three.

8.2 Suggestions from authors

Authors are requested to suggest persons competent to review their manuscript. However, please note that this will be treated only as a suggestion, the final selection of reviewers is exclusively the Editor's decision. The reviewers remain anonymous in any case.

The Editor is fully responsible for decision about the manuscript. The final decision, whether to accept or reject a paper, rests with him/her. The Managing Editor only communicates the final decision and informs the author about further processing.

8.3 Revised manuscript submission

When revision of a manuscript is requested, authors should return the revised version of their manuscript as soon as possible. Prompt action may ensure fast publication, if the paper is finally accepted for publication in .... If it is the first revision of an article authors need to return their revised manuscript within 60 days. If it is the second revision authors need to return their revised manuscript within 14 days. If these deadlines are no met, and no specific arrangements for completion have been made with the Editor, the manuscript will be treated as a new one and will receive a new identification code along with a new registration date.

8.4 Final proofreading

Authors will receive a PDF file with the edited version of their manuscript for final proofreading. This is the last opportunity to view an article before its publication on the journal's web site. No changes or modifications can be introduced once it is published. Thus authors are requested to check their proof pages carefully against manuscript within 3 working days and prepare a separate document containing list of all the changes that should be introduced. Authors are sometimes asked to provide additional comments and explanations in response to remarks and queries from the language and technical editors. In case the authors do not deliver the list of corrections to proofs in the requested time the manuscript will be published as is.

8.5 Reprints

Because the journal is published in an Open Access model, and has no printed version, the authors receive no reprints.

8.6 Erratum

If any errors are detected in the published material they should be reported to the Managing Editor. The corresponding authors should send appropriate corrected material to the Managing Editor via email. This material will be considered for publication in form of erratum in the earliest available issue of ....

8.7 Copyright

9 Paper writing guide

9.1 Paper elements

1. title page with:
   (a) title (short title),
   (b) full name(s) of author(s),
   (c) name and address of workplace(s),
   (d) personal e-mail address(es),
2. abstract,
3. up-to five keywords,
4. text,
5. reference lists.

9.1.1 Abstract

An abstract must accompany every article. It should be a brief summary of the significant items of the main paper. An abstract should give concise information about the content of the core idea of your paper. It should be informative and not only present the general scope of the paper but also indicate the main results and conclusions. An abstract should not normally exceed 200 words. It should not contain literature citations or allusions to the tables or illustrations. All non-standard symbols and abbreviations should be defined.

In combination with the title and key-words, the abstract is an indicator of the content of the paper. Authors should remember that on-line systems rely heavily on the content of titles and abstracts to identify articles in electronic bibliographic databases and search engines. They are therefore requested to take great care in preparing these elements.

9.1.2 Text

9.1.2.1 General rules for writing

- use simple and declarative sentences, avoid long sentences, in which the meaning may be lost by complicated construction;
- be concise, avoid idle words;
- make your argumentation complete; use commonly understood terms; define all non-standard symbols and abbreviations when you introduce them;
- explain all acronyms and abbreviations when they first appear in the text;
- use all units consistently throughout the article;
- be self-critical as you review your drafts.

Figure 1

Fig. 1. A figure caption should be placed **below** the figure.

9.1.2.2 Structure of a paper

Research papers and review articles should follow a strict structure. Generally a standard scientific paper is divided into:

- introduction: you present the subject of your paper clearly, you indicate the scope of the subject, you present the goals of your paper and finally the organization of your paper;
- main text: you present all important elements of your scientific message;
- conclusion: you summarize your paper.

Experimental part and/or calculations should be presented in sufficient details to enable reader to repeat the original work.

9.1.2.3 Footnotes/End-notes/Acknowledgments

We encourage authors to restrict the use of footnotes. If necessary, please make end-notes rather than footnotes. Allowable footnotes/end-notes may include:

- the designation of the corresponding author of the paper;
- the current address of an author (if different from that shown in the affiliation);
- traditional footnote content.

9.1.2.4 Tables

Authors should use tables only to achieve concise presentation, or where the information cannot be given satisfactorily in other ways. Tables should be numbered consecutively using Arabic numerals and referred to in the text by number. Each table should have an explanatory caption which should be as concise as possible.

9.1.2.5 Figures

Authors may use line diagrams and photographs to illustrate theses from their text. The figures should be clear, easy to read and of good quality. Styles and fonts should match those in the main body of the article. All figures must be mentioned in the text in consecutive order and be numbered with Arabic numerals.

Figure 2

Fig. 2. A figure caption for Fig. 2.

9.1.2.6 Typesetting

Type main text in roman (upright) font. The chemical symbols and compounds, units of measure, most multi-letter operators and functions should are written in roman upright as well. The variables, constants, symbols for particles, most single-letter operators, axes and planes, channels, types (e.g., n, p), bands, geometric points, angles, lines, chemical prefixes, symmetry designations, transitions, critical points, color centers, quantum-state symbols in spectroscopy, and most single-letter abbreviations should be written in roman italic. Boldface roman type is reserved for indicating vectors and in some special cases matrices.

9.1.2.7 Mathematical symbols

The multiplication signs are reserved for a vector product ($\mathbf{A} \times \mathbf{B}$) and simple dot product ($\mathbf{A} \cdot \mathbf{B}$). The only exception are numbers expressed in scientific notation (9.7×10^3 MeV).

9.1.2.8 Units

Units and dimensions should be expressed according to the metric system and SI units. This system is based on: meter (m), second (s), kilogram (kg), ampere (A), kelvin (K), mole (mol), and candela (cd). Most units are spaced off from the number, e.g. 12 mV. The only exceptions are:

$$1\%,\ 1‰,\ 1°\mathrm{C},\ 1°,\ 1',\ 1''.$$

Decimal multiples or sub-multiples of units are indicated by the use of prefixes

$$\mu{=}10^{-6},\ \mathrm{m}{=}\ 10^{-3},\ \mathrm{c}{=}\ 10^{-2},\ \mathrm{d}{=}\ 10^{-1},\ \mathrm{da}{=}\ 10^{1},\ \mathrm{h}{=}\ 10^{2},\ \mathrm{k}{=}\ 10^{3},\ \mathrm{M}{=}\ 10^{6},\ \mathrm{G}{=}\ 10^{9},\ \textit{etc.}$$

Compound units are written as

$$4221.9\ \mathrm{J\ kg^{-1}\ K^{-1}} \text{ or } 4221.9\ \mathrm{J/(kg\ K)},$$

with a thin space between unit parts.

Authors should indicate precisely in the main text **where tables and figures should be inserted**, if these elements are given at the end in the original version of the manuscript (or supplied in separate files). If this information is not provided along with the manuscript, we will assume that the figures and/or tables should be insert at the closest position to first reference to them in the published paper.

9.1.2.9 Multimedia and images

Authors can attach files in most popular formats, including (for example):

- images in BMP, GIF, JPEG formats,
- multimedia files in MPEG or AVI formats.

However please keep to file types that are read by standard media players (e.g. RealPlayer, Quicktime, Windows Media Player) and/or standard office applications (Adobe Acrobat Reader, Microsoft Office etc.).

Your attachments may be accessible through links to external locations or to our internal locations (if you choose the second option, please remember to send us your attachments).

Please remember that your images, video and animation clips are intended for Internet use and we need to consider the needs of users with slow Internet connections. Please try to minimize file sizes by using a lower resolution or number of colors for images and animations (as long as the material is still clear). To help you in formatting your images (including tables and figures) or multimedia files, please submit your paper with separate attachments, which are used in your paper.

9.1.2.10 English language

Journal is published only in English. Make sure that your manuscripts are clearly and grammatically written. Please note that authors who are not native-speakers of English can be provided with help in rewriting their contribution in correct English. Try to prepare your manuscript in an easily readable style; this will help avoid severe misunderstandings which might lead to rejection of the paper.

9.1.3 Reference list

A complete reference should give the reader enough information to find the relevant article. All authors (unless there are six or more) should be named in the citation. If there are six or more, list the name of the first one followed by “et al”. Please pay particular attention to spelling, capitalization and punctuation here. Complete-

ness of references is the responsibility of the authors. A complete reference should comprise the following:

9.1.3.1 Reference to an article in a journal

Elements to cite: Author's Initials. Surname, – if more authors, see examples below, Title of journal – abbreviated according to the ISI standards[17], volume number, page or article number (year of publication). Please supply DOI or URL for e-version of the papers. See Refs. [1–8] for example.

9.1.3.2 Reference to a book

Elements to cite: Author's Initials. Surname, Title, Edition – if not the first (Publisher, Place of publication, Year of publication) [9].

9.1.3.3 Reference to a part/chapter book

Elements to cite: Author's Initials. Surname, In: Editor's Initials. Editor's Surname (Ed.), Book Title, Edition – if not the first, (Publisher, Place of publication, Year of publication) page number [10].

9.1.3.4 Reference to a preprint

Elements to cite: Author's Initials. Surname, arXiv:preprint-number and version [11, 12].

9.1.3.5 Reference to a conference proceedings

Elements to cite: Author's Initials. Surname, In: Editor's Initials. Editor's Surname (Ed.), Conference, date, place (town and country) of conference (Publisher, place of publication, year of publication) page number [13].

9.1.3.6 Reference to a thesis

Elements to cite: Author's Initials. Surname, D.Sc./Ph.D./M.Sc./B.Sc. thesis, University, (town, country, year of publication) [14].

9.1.3.7 Reference to an article in a newspaper

Elements to cite: Author's Initials. Surname, Newspaper Title, date of publication, page number [15, 16].

17 http://images.isiknowledge.com/WOK46/help/WOS/0-9_abrvjt.html

9.1.3.8 Reference to a patent

Elements to cite: Originator, Series designation which may include full date [17].

9.1.3.9 Reference to a standard

Elements to cite: Standard symbol and number, Title [18, 19].

Please add language of publication for materials which are not written in English. Indicate materials accepting for publications by adding "(in press)". Please avoid references to unpublished materials, private communication and web pages.

You should make sure the information is correct so that the linking reference service may link abstracts electronically. For the same reason please separate each reference from the others.

Before submitting your article, please ensure you have checked your paper for any relevant references you may have missed.

9.1.4 Submission formats

Manuscripts for ... should be submitted in the LaTeX format with figures in EPS, PDF or PNG format. Authors are strongly encouraged to register their manuscript in arXiv preprint server and submit it to our Editorial Manager using arXiv's paper ID.

9.1.5 Supplementary data

You can also submit any supplementary data files as well. These may include long tables (in HTML or plain TXT format) or movies (preferably in AVI format).

References

[1] A. P. Raposo, H. J. Weber, D. E. Alvarez–Castillo, M. Kirchbach, Cent. Eur. J. Phys. 5, 253 (2007)
[2] J. Barth et al. (SAPHIR Collaboration), Phys. Lett. B 572, 127 (2003)
[3] S. Chekanov et al., Eur. Phys. J. C 51, 289 (2007)
[4] K. Malarz, Postepy Fizyki 57, 235 (2006) (in Polish)
[5] G. Meng, Cent. Eur. J. Phys., DOI:10.2478/s11534-007-0038-1
[6] R. Hegselmann, U. Krause, Journal of Artificial Societies and Social Simulation (2006), http://jasss.soc.surrey.ac.uk/9/3/10.html
[7] A. Dybala, Cent. Eur. J. Chem. (in press)

[8] A. Dybala, Przeglad chemiczny (in Polish, in press)
[9] M. Lister, Fundamentals of Operating Systems, 3rd edition (Springer-Verlag, New York, 1984)
[10] C. K. Clenshaw, K. Lord, In: B. K. P. Scaife (Ed.), Studies in Numerical Analysis (Academic Press, London and New York, 1974) 95
[11] M. Majewski, K. Malarz, arXiv:cond-mat/0609635v2 [cond-mat.stat-mech]
[12] J. A. C. E. Solano, arXiv:0707.1343v1 [astro-ph]
[13] A. Kaczanowski, K. Malarz, K. Kulakowski, In: T. E. Simos (Ed.), International Conference of Computational Methods in Science and Engineering, Sep. 12-16, 2003, Kastoria, Greece (World Scientific, Singapore 2003) 258
[14] A. J. Agutter, Ph.D. thesis, Edinburgh University (Edinburgh, UK, 1995)
[15] A. Sherwin, The Times, Jul. 13, 2007, 1
[16] M. Dzierzanowski, Wprost, Jul. 8, 2007, 18 (in Polish)
[17] Philip Morris Inc., European patent application 0021165 A1, Jan. 7, 1981
[18] ISO 2108:1992, Information and documentation — International standard book numbering (ISBN)
[19] ISO/TR 9544:1988, Information processing — Computer-assisted publishing — Vocabulary