How can text generation tools be used within the different legal professions? (a background analysis by Peter Homoki used in the AI4Lawyers project)

1.1.1 Introduction

What is natural language generation? Natural language generation (NLG) is usually presented as one of the applications of natural language processing, contrasting it with text analytics[1]. The contrast is the following: if the goal of automated text analysis is to draw automated inferences from a specific natural language text (text surface), including from any level of grammatical structure, content, meaning or even the intention of the author of the text), then the goal of **natural language generation** is the opposite: to generate from some computer data (representation) a text or speech that sounds as natural as possible or as human as possible.

Within the problem of language generation, we will focus on the problem of text generation, since the automatic generation of speech from text has few legal specifics (similar to the conversion of speech into machine text). This is not to say that there is no research to be done in this area, but such research would be a rather narrow cross-section of, for example, forensic rhetoric of a given language and text to speech.

For the average user, the best-known uses of text generation are probably machine translation[2] and various chatbots and virtual assistants[3]. However, the range of uses of text generation tools is much wider than this. Text generation includes word-processing applications or certain bulk emailing solutions, template-based creation of contract documents for large numbers of customers and supporting such workflow (usually known as "document assembly" or sometimes as "computer-aided drafting"[4]), or applications of a completely different nature, such as the producing naturalistic-looking text forecasts from incoming weather data, or reports from financial or military data, etc.[5]. Question-and-answer type systems are also included in this category, but in such systems, the focus is on finding the correct information, given the expected short answers – still, this is also a text generation task[6]. Likewise, this category of text generation also includes solutions aimed at summarising larger texts, trying to extract the essence of the larger text (text summarization[7]).

To a large extent, the architecture of text generation solutions depend on the application domain: there are significant differences in their technical requirements and the economics of such solutions. The quality requirements for an ordinary chatbot are completely different to a text report for medical professionals: while the first may at worst become a PR experiment going bad, the lives of many people may depend on the latter. Similarly, the optimal architecture

---

[1] Nitin Indurkhya and Fred J Damerau, *Handbook of Natural Language Processing* (2., Chapman & Hall/CRC 2010) 5.

[2] Daniel Jurafsky and James H Martin, *Speech and Language Processing* (3rd draft, 2020) 203–230 <https://web.stanford.edu/~jurafsky/slp3/ed3book_dec302020.pdf>.

[3] ibid 492–525.

[4] Marc Lauritsen, 'Current Frontiers in Legal Drafting Systems' (2007) 1 <https://static1.squarespace.com/static/571acb59e707ebff3074f461/t/5946f1e39de4bb69d253380c/1497821669156/CurrentFrontiers.pdf>.

[5] Indurkhya and Damerau (n 1) 530–555.

[6] Jurafsky and Martin (n 2) 464–490; Indurkhya and Damerau (n 1) 485–510.

[7] Jurafsky and Martin (n 2) 197. For a contemporary Hungarian solution, see Zijian Győző Yang, Attila Perlaki and László János Laki, 'Automatikus Összefoglaló Generálás Magyarnyelvre BERT Modellel' (2020) 16 Magyar Számítógépes Nyelvészeti Konferencia 343 <http://acta.bibl.u-szeged.hu/id/eprint/67658>; Ádám Feldmann and others, 'HILBERT, Magyar Nyelvű BERT-Large Modell Tanítása Felhő Környezetben', *XVII. Magyar Számítógépes Nyelvészeti Konferencia* (Szegedi Tudományegyetem TTIK, Informatikai Intézet 2021) <http://real.mtak.hu/id/eprint/120856>.

of a text generation solution to properly identify a possible set of questions of a few hundred elements and then provide a few sentences of answer is very different from the design of a solution where the possible answer is of several pages of length. Likewise, the economics of practical use will limit the practical usability of individual solutions. If it were still true today that a sophisticated NLG system would have to generate at least ten thousand pages per year to be worth creating and maintaining in practice,[8] this would also be a severe judgment on the opportunities for millions of users of 'small' languages, not only within the legal domain but in any field of specialisation.

In light of this, it is perhaps worthwhile to attempt, as a practising lawyer, to review from the selfish perspective of a professional what these techniques might be useful for. Such an analysis should take into account the available market solutions, as well as views in the literature. Given that the author is a lawyer, the focus of the question of exploitability will also be on the specificities of this profession, but we will also briefly address some specific requirements of the professions of judge or latin-type notary.

1.1.2 On document assembly software

1.1.2.1 Division of document assembly software in the literature – how a tool using NLG technology can be better than ordinary uses

Document assembly software may be divided into categories according to the point of view of literature or the market.

As early as 1993, Branting contrasted the document-centric approach with the issue-centric approach in the literature on judicial document drafting.[9] In the document-centric approach, Branting emphasised that the focus is on the appropriate selection of relevant document components (e.g. a paragraphs, clauses) and instantiating user-provided values for variables embedded in those components. This consists of templates and rules for filling in the template. On the other hand, Branting defined the other category based on his own proposal, the issue-oriented approach: here, in addition to (above) the presentation layer of the text itself, a presentation layer is created whose function is to evaluate the rules necessary for the judicial decision, to support the decision. Thus, together with the text, the rules evaluated during the judicial decision and the legal predicates (statements) representing the rules are also stored as well, and during the automation of the “text”, all the rules relevant for the given decision are also evaluated.

In a later work the same author (together with other co-authors) highlights the procedural, template-based and the discourse-based approaches[10]. In the case of the procedural approach, instructions defined in an imperative-type programming language select the text elements required for the new document on the basis of specific values chosen by the user for the document to be created. This category included the system named ABF, and even today, the most popular document assembly solutions use languages very close to this one. In the template-based approach, according to the authors, designers create a document class (the “template”) in which they record the common text elements of the instances under the template

---

[8] Ehud Reiter and Robert Dale, *Building Natural Language Generation Systems* (Cambridge University Press 2000) 28–29.

[9] L Karl Branting, ‘An Issue-Oriented Approach to Judicial Document Assembly’, *Proceedings of the 4th International Conference on Artificial Intelligence and Law* (Association for Computing Machinery 1993) 2 <https://doi.org/10.1145/158976.159005>.

[10] L Karl Branting and others, ‘Integrating Discourse and Domain Knowledge for Document Drafting’, *Proceedings of the 7th International Conference on Artificial Intelligence and Law* (Association for Computing Machinery 1999) 199 <https://doi.org/10.1145/323706.323800>.

and incorporate into this text (1) tokens that indicate specific facts about a document (variables, fields) and (2) conditions that determine which text elements should be inserted into a document and in under what conditions. At the time of the publication of Branting's paperfor the authors' main idea was to implement discourse-based document generation software. The approach aimed to capture content beyond the textual surface, which meant capturing information about the structure of the discourse: what the text is about, what the communicator's aim was (illocutionary aim structure, an inference tree expressing subgoal relations among the applicable legal predicates), how this is reflected in the text, and what the rhetorical structure of the text is (where is a contrast between the parts of the text, which one explains the other part of the text, or serves as a background, supports the other, etc. - these are all rhetorical relations that add up to a rhetorical structure).

A further classification in the literature simply contrasts the "mail merge" and NLG-based approaches.[11] The essence of the mail merge designation is that these word processing functions are themselves capable of providing complex document automation functions, not only to specify variable data in the template (e.g. fields), but also to define conditional text elements, loops (e.g., in Microsoft Word, the bracketed fields also under the "Mailings" tab.)

The mail merge function is easy to understand, since almost everyone has such a tool at their disposal, even if not everyone is aware how sophisticated the use of such tools could become. In comparison with mail merge, the advantages of the NLG approach is more difficult to illustrate for the layman, since NLG tools are based on a number of solutions that are also very different from each other, and in any case are accessible only for a more limited audience (e.g. the details of the solutions of Arria or Yseop, which are not widely used or known, operate in a way where the details remain hidden from the end-users of such systems).

The common ground of NLG tools is perhaps best expressed as the common research and scientific background and computational linguistics roots on which such tools build[12]. In contrast, mail merge products focus only on the most immediate challenges of template building. Even mail merge tools may build upon complex, Turing-complete programming languages, but such tools do not focus on or reflect linguistic problems.[13] In In 2016, Reiter also wrote a short blog post regarding the uncertainties of the distinction between NLG and template.[14]

NLG tools are as diverse as those described in the text generation application areas, they can help to run a chatbot or to customise template texts, to transcribe data points into text or to create summaries of large texts, etc. Technically, they can be based on a multilayer neural network architecture (see e.g. the BERT model[15]), but also on any software without machine learning capabilities at all, based on a dictionary and manual rules. An NLG tool can also build on traditional computational linguistic solutions such as part-of-speech analysis to select the word fitting the sentence to be created best from a large number of possible words, or a morphology generator to produce the correct conjugation form.

---

[11] Reiter and Dale (n 8) p. 26.

[12] Indurkhya and Damerau (n 1) p. 121–140.

[13] Reiter and Dale (n 8) p. 26–27.

[14] Ehud Reiter, 'NLG vs Templates: Levels of Sophistication in Generating Text' (September 2016) <https://ehudreiter.com/2016/12/18/nlg-vs-templates/>.

[15] Yang, Perlaki and Laki (n 7).

1.1.2.2 Market-based division of document assembly software

In their review, Markovic and Gostojic highlighted four document assembly software products, HotDocs, ContractExpress, Pathagoras and XpressDox[16]. Of course, there are a lot more tools beyond these, so many and this is such a fast-changing market that it is almost impossible to list them all. (In addition to the four already listed, we just mention Legito, docassemble, ClauseBase, Woodpecker or Juro as examples.)

Looking at the products, we can see products where document assembly support is only a small slice of the functionality offered, the main purpose being rather to manage contracts (the so-called contract lifecycle management tools), including the collection of signatures (e.g. Juro) or even the step of negotiating the contract text, up to events such as monitoring expiry or renewal after the contracting process is completed (e.g. Coupa). In this case, the editing and assembly capability is typically only one of the many features provided, and not necessarily in the focus of product development (in such cases, the vendor itself may not necessarily call and position its product as a document assembly product).

For specific document assembly functionality, the common points are that they have a solution for (1) the design phase, the creation of the template, and (2) the use of those templates. In the design phase, on the one hand, there is also a simple word processing (or importing) part, and a part that turns such text into a template: external conditions, data relations and variables to be filled in are defined, and rules for the inclusion or repetition of the text are specified. In general: by defining business rules (sometimes called "business logic") the relationship between the elements of the text, its variables and the outside world is captured during the process of template authoring. Template authoring may be handled on a dedicated web interface (e.g. Legito), but it may also be handled as a whole by a pure client-side or local server-side solution that is adapted to a specific word processor, or even as a traditional Word extension (e.g. XpressDox, HotDocs Author, Woodpecker). Of course, it is also common for some template authoring functionality to appear partly on a web interface (running on a separate server) and partly on the end-user machine (e.g. ActiveDocs). The implementation of these functions can be extremely complex, regardless of whether the template authoring function is running on a client-side solution or on a cloud-based web interface. Unfortunately, quite often the users can only recognize some serious limitations of the given template authoring tool only after they have invested a very significant amount of time in the given solution. Of course, in such case the products will cause further disappointments to the users when they realize there is no transition, compatibility between templates of different products, there is no standardisation in this field (other than of course, the formatted text itself).

The variables to be initialized also determine what data is to be collected from data sources outside the template. This data source can be answers given by users by way of a user interface, but can also be data sources linked to external programs. UI-derived questions are traditionally referred to as interviews in this context. Depending on the product, the questions asked may be inferred from data fields mostly automatic (XpressDox) or can be controlled in very much detail if needed (e.g. HotDocs), including the provision of explanations or graphical elements used in the interview, validation rules for the data entered etc. Typically, the answers given during the interview can be saved to generate similar documents again.

The functionality offered by such products is also fundamentally influenced by the primary target audience. If the target audience is a company where any technical integration of such

[16] Marko Markovic and Stevan Gostojic, 'Knowledge-Based Legal Document Assembly' (2020) abs/2009.06611 CoRR, p. 2 <https://arxiv.org/abs/2009.06611>.

products is usually subject to a separate project with budget and experts, then obviously much more sophisticated integration solutions can be built. If the editing and filling in of templates is the task of a dedicated employee or consultant, then even the term “simple template creation” will have a very different meaning compared to what can be generally expected from a practising lawyer or his assistant trying their hands at such solutions. For some products, it is acceptable to use IT template engines or programming languages existing in the market during the process of document automation (e.g. jinja2, Python for docassemble, or Javascript-based implementation of more complex interview functions). Obviously, these latter solutions allow for a wide variety of solutions, but with such products, end-users can only be given access to ready-made templates. With such less user-friendly (user targeted) tools, one may not expect end-users to edit the templates themselves. However, this template authoring part is the most complex part of the document assembly process, and therefore this is the central objective of products such as ContractExpress, HotDocs or XpressDox.

Most of the products available on the market are language agnostic or at most, adapted to a few languages which adaptation cannot be extended by the users. For example, one may find easy-to-implement built-in language-specific solutions such as respecting regional formatting rules, dates and decimal comma/decimal point issues in calculations (see e.g. HotDocs/LANGUAGE), but even language support for proper verb and noun conjugations (e.g. ClauseBase for English, French, German and Dutch conjugation or declination, or in case of replacement, aligning nouns previously written in masculine to the new feminine form, etc.).

Another important difference is that some products are only available as a cloud service (e.g. ContractExpress, Legito), which also means that if the subscription is cancelled, all the energy invested in template authoring is wasted (as already menitoned, there is no way to exchange formats between different products). The advantage of cloud-based solutions is that users can get up and running faster and there are no significant upfront IT implementation costs. In the case of cloud solutions, providing integration is a more critical issue due to the possibility of data connections used. Unless the provider of the cloud product makes appropriate APIs connections available, there is not much a user can do, unlike installed (on-premise) solutions where deeper integration can be worked out between running processes and at operating system level. So in the case of cloud solutions, it is entirely up to the vendor to define whether to provide an API at all (e.g. Juro) and if so, for which functionality they provide this access. Without appropriate APIs, export functions and standardised, compatible operation between different products, cloud solutions can loose much of their appeal.

As regards pricing of products on the market, the range is very wide, from free but non-user focused products (such as docassemble) to enterprise pricing that is not public. There are standalone solutions for small users, which can be purchased for around $400 per year (e.g. WoodPecker, XpressDox), and for most of the installable (on-premise, not cloud) solutions, annual version upgrades are no longer required for use, so in principle only a one-off licence fee is payable.

1.1.3 Instead of divisions based on literature: what features can be added to the document assembly solutions?

Let's look at the areas where the functionality of assembly solutions can be extended - first by reviewing the reasons and justifications for the distinction in the literature, and then by considering what extra-textual functionalities may exist in such tools.

The works cited above, by making these divisions, have mostly attempted to point out that there are specific, linguistic-based software functionalities beneficial for a particular use of natural text generation (as if the works cited were trying to justify their own existence by defining such

categories.) The procedural and template-based approaches have essentially come into existence simultaneously with computer text processing capabilities (computer macro-languages, template engines, text insertion capabilities based on database access or variables, etc.), even in such tools there are symbols in the raw text to give instructions on how to process or use the text.

Let us take a look at what the authors cited in the literature highlight as the advantages of NLG solutions As an advantage of his approach, Branting highlighted the flexibility of the method, it can be used to support judges' decisions, it also helps to explain the decision and it is easier to maintain.[17] In a later study, he pointed out that for certain uses, it is also necessary for the user to be able to see at any time why the text was included in the draft, what caused the change, and to be able to modify it if necessary ('queriable liveliness'[18]). In his study at the time, he criticised commercially available tools for being linguistically and legally (that is knowledge) independent solutions, which made their use inflexible. Already at that time, the author emphasised how much NLG research could help in this area, quoting from an earlier work by Reiter: easier text maintainability, easier to make changes and updates over time, all of which could result in better quality, more comprehensible texts, with document parts that are able to reflect and follow relationships that span across sentences (e.g. the rhetorical structure referred to above), making them easier to adapt to several languages or even to check for compliance purposes.

A year later, Reiter and Dale, when comparing machine and human text generation in general (i.e. not comparing template-based and NLG-based document composition solutions), highlighted essentially the same advantages:[19] text generation solutions can provide greater consistency than human ones, produce more standardized output faster and in more languages. However, the same study also points out that the development of NLG solutions pays off only if they are used to produce a sufficiently large number of documents[20] – obviously, the cost of developing each solution has changed since then, but this aspect should not be ignored and is still important today.

In terms of the source of the benefits listed, it is fundamental that, in the case of discourse-based and issue-centred solutions, in addition to (and beyond) the 'raw text', authors may wish to capture additional information as well that, in a given use case, may be of further benefits.

The issue- or discourse-based approach is a judge-focused use case; in another study, for similar purposes, the set of legal rules represent described by IT tools define the questions that the user has to answer during the interview.[21] Obviously, such a system or parts of it (e.g. the knowledge base of a machine-based description of rules) is not appropriate unchanged to be used by, say, lawyers or other users. Similarly, the conditions for making inferences and deviating from machine-made inferences are different in the case of automated decision making by a government body than in the case of a criminal conviction. Each of the solutions cited builds on some kind of data beyond the text: be it a set of rules captured in a machine-processable

[17] Branting (n 9) 3.

[18] Branting and others (n 10) 2–3.

[19] Reiter and Dale (n 8) 29–30.

[20] ibid 28–29.

[21] Markovic and Gostojic (n 16).

form, a rhetorical structure (DocuPlanner[22]), some semantic descriptive content or metadata related to the subject matter of the text, or even a more general description of legal knowledge[23].

In his general theoretical outline of systems supporting document production, Lauritsen also mentions the representability and capture of a number of factors beyond the text,[24] so that, in addition to the semantic elements mentioned above, he includes, for example, explanations, the source of the text, conditional or repetitive texts, mandatory and possible schema elements, etc., and also mentions an approach to include a way to unambiguously reference texts within a complete legal practice or external references (court cases etc.) as a similar factor.

Thus, in terms of describing the capabilities of document assembly tools, one can find technical features that are independent of language and a legal knowledge (such as the issue of inserting text that is language and and language domain agnostic, or the use of variables and database relations), but this technique has been available since the beginning of machine word processing, as already mentioned, it is essentially contemporaneous with the computer, and has even been made available to lawyers as a dedicated market product since at least 1979 (ABF), most word processors also provide such features in a ready-made form.

However, the specific process of generating text can be supported in many areas beyond this. The question is whether in such a case a different product is needed for the different use cases. If that is so, this significantly reduces the likelihood document assembly products becoming widely used solutions.

The more versatile these solutions are designed to be, the more complex the automation itself will be to implement and to use. This is not simply a matter of whether or not a user interface builds on contemporary graphic design traditions and user interface standards or it was made thirty years ago. The problem is that there are very complex decisions behind the assembly of a document, and such decisions simply cannot be captured in a transparent yet still flexible way. Compare e.g. Legito's template editing interface when used for complex multi-conditional paragraphs relying on multiple variables and interview items, when having to loop over several documents: the UI is definitely more modern with this tool than what authoring solutions originating from 1990s look today, but the solution will still be complex, the authors will still need dozens of hours of training – the complexity is still there, but in a rather different way. More generally, experimenting with “no code” solutions could help many lawyers overcome their intial fears. You can just click on a diagram and select the solutions you need (see e.g. solutions like Google Blockly). However, this will not solve the inherent complexity in how documents are drafted: the task does not get much easier by not having to remember and type one-word commands, but instead dragging and stacking blocks being a visual representation of the same statements. Such complexity at the document component level is not a problem stemming from how information or possible commands are represented to the user (a problem of the user interface), it is more a problem resulting from lacking proper abstraction and modularity. Current market products marketed to small law firms do not have the right components, because their focus is on inserting or looping text English language text. However, even if the output of a lawyer is often a text, the text is just the final output of higher level processing based on legal knowledge domains and linguistic mental tools.

---

[22] Branting and others (n 10).

[23] Imre Kilián, ‘Szemantikai Alapú Jogi Tudás-Menedzsment Technológiák’ (phdthesis, Pécsi Tudományegyetem Állam- és Jogtudományi Kar, Doktori Iskola 2013) 150–156 <https://ajk.pte.hu/files/file/doktori-iskola/kilian-imre/kilian-imre-vedes-ertekezes.pdf>.

[24] Lauritsen (n 4) 10.

There are considerable constraints on what kind of wording is acceptable in legal texts and if the linguistic layer of the template text is not sufficiently abstract, then e.g. business logic tools will have to be used for linguistic corrections. This will slow down the creation of the templates, increase the cost of automation. If the legal professionals are expected to cope with this complexity, this will inevitably push such users away from these solutions. In many cases, it is not sufficient to think in terms of templates at the contract level, but in terms of "clauses" as general as possible, since in this case the provisions can be reused regardless of the type of contract. Obviously, where possible, similar legal objectives should be addressed by using similar wording (for example, if the same payment terms are used in both an agency and a sales contract, it should be possible to automate this so that the same wording can be used for parties called principal, agent, seller and buyer, etc.). This requires a rewording of the input text, making it more general, and this may be very difficult to achieve in many languages.

The cost of automation is further increased if, in addition to the linguistic layer, we want to capture knowledge-related information in the document assembly solution, such as the legal knowledge base, the legal or other semantic content behind the text. Very few people have experience in capturing and describing legal knowledge, and each such person's experience is typically based on a very different product.

So, unfortunately, the more versatile the use, the higher the cost of automation are. The cost of automation is therefore increased by narrow user base, the lack of standardisation and of data exchange capabilities between products, and finally, the different needs from language to language. Unfortunately, until this standardisation is achieved and the market for smaller users is consolidated, the costs of automation in the legal industry will not decrease. Without such support, it may still be the case that is not worth supporting and automating the assembly of a document for a particular use case.

1.1.3.1 On how text summarization tools and multi-layer neural models in general can be used by legal professionals

The range of tools used for text generation is not limited to document assembly products, so it is also worth briefly discussing the potential of text extraction tools for legal uses. The specific study under review concerned use in Hungarian, but this is an example of how similar results can be expected for other languages of similar size.[25] In this case, the generated text is based on a so-called end-to-end neural model, i.e. it does not replace only some components of a classical natural language processing pipeline, but all of it.

The cited study showed the results of using a multilingual version of BERT for a given small language to produce a few-sentence summary of some articles in Hungarian. To simplify to the extreme what they did in the study: BERT is a language model architecture based on a multilayer neural network developed and published by Google, which is used in a wide variety of ways for language machine learning tasks. One of the possible BERT models, already pre-trained by Google for multiple languages, has been reused by the researchers (BERT-Base, Multilingual Cased, elsewhere referred to as multi-BERT). The researchers fine-tuned this model for text summarization tasks and experimented with which architecture was giving the best results. In the case of the so-called extractive model, the sentences of the article to be summarised (their numerical representation) were fed as input, and the model was tasked with correctly guessing the three sentences that best described the content, and then, as part of the training, they compared the final output of the model with the handwritten summaries, and thus improved the modelling by providing feedback. The final result provided "hit rate" of 55.46%

[25] Yang, Perlaki and Laki (n 7).

(the similarity of words, more precisely unigram coverage based on ROUGE-1 method). The later research the results of which were also published in English show different results based on different models and corpus (using an abstractive model[26]). The extent to which such results can be used for legal purposes will be discussed later.

Other researchers have shown that in certain use cases, better results can be achieved if, instead of using the published Google model trained on several languages, an independent training is carried out according to the same method, based solely on the given (in this case also Hungarian) language corpus and text material (with new pre-training). One research presented the teaching of a smaller model (huBERT[27]), another research detailed the training on a larger text size (HILBERT[28]). At current prices, this latter pre-training costs about $6500 and required 24 GB of input data for training. The model trained on exclusively the Hungarian language corpus provided significantly better results than the multi-language version.[29] It's interesting to note that for a layman's (lawyer's) point of view, it is surprising how could texts and results can be generated by using a multi-language model.

However, while using a model trained on a (generic) Hungarian corpus, can yield significant improvements compared to the multi-language (generic) model, models trained specifically on legal corpus can only bring far more limited improvements. A study has been published which shows whether greater accuracy (better results) can be achieved in certain tasks if the pre-training itself is done on legal texts, compared to only fine-tuning on some legal text (LegalBERT[30]). Pre-training or fine-tuning was based on 1.9 GB of EU legal acts (EURLEX), 0.6 GB of court decisions and other similar data, e.g. the US Securities Exchange Commission's EDGAR database (3.9 GB). The survey was performed on text comprehension tasks of a legal nature such as who is the contracting party, what is the title of the contract, what is the governing law or jurisdiction, or what are the key elements of the lease in the text (e.g. who is the landlord, termination date, fixed term, rental rate, etc.[31]). The results showed a slight improvement, and the more difficult the task is, the greater this improvement was, but on average the maximum improvement did not exceed of 2.5%. It should be added that for these legal tasks, it is often not the BERT model that gave the best results, but other multilayer neural network architectures (e.g. for lease contracts, it found the correct provision 87% of the time).

For us, this is only illustrative: solutions based on multilayer neural network models can be a promising solution for legal text summarisation, but their use in a small number of languages in a given Member State requires further investment and research and development. The investment may mean training on legal corpus in a given language, which requires preparation of legal texts for training. But even training itself does cost. There are also other prerequisites such as a developed teaching kit for teaching, which also requires human input (see e.g. 2080+289 samples for training on recognising governing law, 2552 samples for the leasehold, etc.[32]). Generally, the size of the available body of law should not be a problem (for comparison,

---

[26] Feldmann and others (n 7).

[27] Dávid Márk Nemeskey, 'Introducing HuBERT', *XVII. Magyar Számítógépes Nyelvészeti Konferencia* (Szegedi Tudományegyetem TTIK, Informatikai Intézet 2021) <http://real.mtak.hu/id/eprint/120856>.

[28] Feldmann and others (n 7).

[29] Nemeskey (n 27).

[30] Ilias Chalkidis and others, 'LEGAL-BERT: The Muppets Straight out of Law School' (2020) abs/2010.02559 CoRR <https://arxiv.org/abs/2010.02559>; Ilias Chalkidis and others, 'Neural Contract Element Extraction Revisited' (2021) abs/2101.04355 CoRR <https://arxiv.org/abs/2101.04355>.

[31] Chalkidis and others, 'Neural Contract Element Extraction Revisited' (n 30).

[32] ibid.

with an average length of about 170 000 anonymised Hungarian court judgments, the size of the Hungarian judicial corpus exceeds the size of the available EU judgments, and that's we can at least suppose that this should not be a problem in other countries with similar size of legal corpus).

However, the size of the available data may still be a serious practical problem for certain uses. Obviously, the SEC's EDGAR contract dataset is simply not available in small languages even if every company and lawyer was able to include all their own contracts into a common corpus (which would of course not be a good idea, as very sensitive information can be decoded from language models, see Carlini2020). Another possible warning worth paying attention to is the trend that in certain important applications, machine learning research is achieving better results by building on ever-larger models. This not only implies an increase in the number of parameters, but also an increase in the amount of training data needed to teach them. This was the case, for example, with GPT-3, which was taught by 45 TB of data. Obviously not only has there never been such a size of Hungarian text or other non-global human-generated text material, but we can expect that probably there will never be such amount (just to gauge the orders of magnitude: the English full Wikipedia, non-compressed and without images, is currently around 70 GB in size and contains 6 362 952 articles, while the Hungarian Wikipedia contains 491 227 articles, etc.). The trend has been followed by other models as well, although information on the size of the teaching material is not always provided (I would rather not draw such a conclusion from the parameter model number per se, but the 175 billion parameter number of the GPT-3 language model was last exceeded by Google Switch with 1.6 trillion parameters[33]).

One study has reported that it takes about 10-100 million words of training material for a model to learn most of the syntactic and semantic features of a language, but a much larger amount of material is needed for the model to recognize other background knowledge that is also present in the language, such as what is usually classified as "common sense"[34]. This underlines the crucial role of language models in solving legal language tasks, and why for many tasks, especially text generation, the appropriate design cannot be left to GPT-3-like models alone.

Even within the same language, the use of legal language may vary (e.g. Austrian-German legal language differences), and it is unrealistic to have models taught on such GPT-3 sized datasets even in medium-sized languages, especially not for training within the legal domain.

Within the legal domain, linguistic accuracy is a high priority, but that is not common to all applications of NLP, so legal use of NLP has to take that special requirement into account.

For legal use, there is also a big difference in the level of required performance based on how the results will be used, e.g. whether the clients will receive the end results created (after being revised by the lawyer) or the results serve only an internal step within the work of the lawyer. If the task is summarising the content of a large number of contracts or extracting elements of them, and the risk is accepted by the client in exchange for a quicker or cheaper survey, then a recall rate of 70% might be sufficient (but e.g. a 60% may not). But we have to be mindful that many clients may probably not wish to outsource these kind of high risk, automated type of uses and rather do it in house if possible, because that's even cheaper – they engage a lawyer so as to minimize the risks as close to zero as possible. If a solution used for text retrieval only

[33] William Fedus, Barret Zoph and Noam Shazeer, 'Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity' (2021) abs/2101.03961 CoRR <https://arxiv.org/abs/2101.03961>.

[34] Yian Zhang and others, 'When Do You Need Billions of Words of Pretraining Data?' (2020) abs/2011.04946 CoRR <https://arxiv.org/abs/2011.04946>.

supports the work of the lawyer and can show the expected results in context to the user before they are accepted as results (e.g. Kira Systems highlighting the location of the provision found in context of the full text), a search result with 80% accuracy can also be of great help. Considering that clients of law firms are often a particular use case

However, for document assembly use, this kind of generous approach might be more problematic or even unacceptable: for a system that produces text from medical data it similarly a considerable problem if an end-to-end neural network model results in displaying a similar sounding, but different disease than the actual one.

If the generated text is short and is viewed in the workflow by a human sufficiently trained to detect the error, minor errors are not a problem. But if the generated text is a long one, and the potential source of the error can appear in countless text locations, then the whole text automation solution could lose its meaning. It is equally dangerous if the error can only be detected by a person who understands the legal-economic context, say in the case of mistakenly exchanging landlord and lessee in the text of a contractual obligation.

As Ehud Reiter has described in his blog posts on the subject,[35] for text generation tasks, models based on techniques that build natural language text directly from non-linguistic data using neural networks are at a disadvantage in performance compared to neural models performing some well-defined tasks within a natural language text generation pipeline.[36]

1.1.3.2 Which tools should be explored or developed in countries using smaller languages?

As you can see above, certain research and development tasks are of interest to a wide range of legal users, while some other tasks are of specific interest only to limited groups.

However, all uses will equally be assisted by the availability of as much data of a sufficiently anonymised nature as possible. It is not important whether these are laws, decisions or contracts, and if decisions, whether they were made by court or by public authorities, whether they are in force or not, of theoretical importance or relevant only for a particular situation. Such a survey of the possible scope of data available is a first step in estimating what kind of legal AI/NLP uses could be feasible and at what cost. This is a prerequisite for all further work, whether this future work is about producing raw training material (e.g. input data set for legal-specific language models) or as input only for the production of manual training material that in turn will be later used for to create models. Despite the risk of re-identification, it would also be useful to have a large database of contract texts, including state and municipal contracts.

Undoubtedly, legal uses would also benefit greatly from extensive support for research in computational linguistics in the official language of the given country, and there is a particularly strong need for this in the field of text generation, so that standard solutions are available for use by all (see e.g. the example of SimpleNLG, https://github.com/simplenlg/). Of course, it's not that legal practitioners themselves should use these tools directly, but making such tools available for the IT companies interested in serving legal practitioners would decrease their costs and increase the probability of them serving practitioners. Supporting standardised solutions could mean the development or extension of the range of open source tools available, the extension of individual functionality of such tools, adapting existing tools to new languages,

---

[35] Ehud Reiter, 'Generated Texts Must Be Accurate!' (September 2019) <https://ehudreiter.com/2019/09/26/generated-texts-must-be-accurate/>; Ehud Reiter, 'Academic NLG Should Not Fixate on End-to-End Neural' (December 2020) <https://ehudreiter.com/2020/12/01/dont-fixate-on-end-to-end-neural/>.

[36] Thiago Castro Ferreira and others, 'Neural Data-to-Text Generation: A Comparison between Pipeline and End-to-End Architectures' (2019) abs/1908.09022 CoRR <http://arxiv.org/abs/1908.09022>.

or even just decreasing the costs of adaptation of such tools by providing new ways of integration, wrapper functions etc. IT companies specialising in serving legal practitioners by e.g. CRM and practice management software offerings are not necessarily NLP specialists, so even merely calling their attention to the availability of such functionalities could be of help.

Although not a gap related to language generation, but for the legal NLP tools available in smaller languages it could make a big difference if some dataset for training data is publicly available in a given language, which training date otherwise requires manual processing. Such could be, for example, the already cited training set for identifying the content of general contractual provisions or specifically the lease and sale contract specific elements[37]). A large body of anonymised judgements could also be very useful in the creation and proliferation of such datasets.

In terms of both text generation and text analysis, significant improvements could be achieved in a number of areas by developing additional template texts in more detail. For example, one such area could be the development of typical clauses (text modules) for individual court submissions by the lawyer (such as motions and briefs etc.), similar to the solution of Casetext's Compose product (https://compose.law/), where typical wording for certain types of motions (e.g. exclusion of expert witness testimony etc.) has been developed in a meticulous editorial effort from jurisdiction to jurisdiction and for specific legal areas. On the one hand, this huge effort helps with text generation tasks, because there is a standardised text the appearance of which needs to be adapted to the context, but at the same time the placement of such a text captures essential information about the higher level structure of the pleading or decision and facilitates the search for decisions on similar motions. Such a use is beneficial not only for lawyers but also for judges and notaries.

It is hard to imagine that word processors can be removed from the centre of use by legal professionals, but it would be useful to support the ability of ordinary users (practitioners) to capture details beyond the text while drafting, such as the semantic content associated with the text. This would be equally useful for contracts and pleadings, decisions and clause repositories. The data thus captured during authoring could then be used to generate syntactically correct text with sufficient generality (e.g. knowledge representation in RDF, and text generation/realisation using RDF-to-text techniques).

Some text generation tools are undoubtedly of marginal relevance for internal legal use, e.g. question-and-answer type systems or most chatbots. Text extraction however, is useful not only for lawyers, but also for prosecutors, police (or even court) staff who need to process large amounts of data in a short time. As regards text generation tools, document assembly solutions are those that could help almost all professions, but with a slightly different focus and user interface, in the automation of other texts and provisions.

Unfortunately, even forty-two years later, the question posed at the American Bar Foundation in 1979 remains as relevant as ever: “Can a lawyer, without the help of a programmer, create and maintain a computer system that helps conduct an interview, draw legal conclusions and assemble the appropriate legal form? Can multiple lawyers use such a system together?”[38]

Without the tools and capabilities (support) described above, the digital divide within the legal professions in the EU will probably be more visible based on the country of the legal professional involved. For many commercially available products, we can see that a solution

[37] For a sample, see Chalkidis and others, ‘Neural Contract Element Extraction Revisited’ (n 30).

[38] James A Sprowl, ‘Automating the Legal Reasoning Process: A Computer That Uses Regulations and Statutes to Draft Legal Documents’ (1979) 4 American Bar Foundation Research Journal 1, 5 <http://www.jstor.org/stable/828045>.

available in a lesser language is not equivalent to an English language solution, and this language gap will only widen, with key competitive tools not tailored to the specific language of the country being used. If these gaps, especially in language generation, cannot be closed, it is not the linguistic science of the country that will lose the most directly, but the ordinary users of the law, whether we are talking about the legal professionals or the public seeking justice.

Of course, for residential use, this gap may not be visible for a very long time, preserving the exclusivity of a particular small language (especially in family law, inheritance law). But in the business field there are already numerous opportunities for parties to choose, purely for reasons of convenience, a contract language, and with it even a governing law, court or arbitration court that is supported by a much wider range of tools and IT services.

I am grateful to Gergely Márk Csányi, NLP engineer, who, after having read the manuscript, provided valuable advice to make it more accurate and professional.

1.1.4 Works cited

Branting LK, 'An Issue-Oriented Approach to Judicial Document Assembly', *Proceedings of the 4th International Conference on Artificial Intelligence and Law* (Association for Computing Machinery 1993) <https://doi.org/10.1145/158976.159005>

Branting LK and others, 'Integrating Discourse and Domain Knowledge for Document Drafting', *Proceedings of the 7th International Conference on Artificial Intelligence and Law* (Association for Computing Machinery 1999) <https://doi.org/10.1145/323706.323800>

Chalkidis I and others, 'LEGAL-BERT: The Muppets Straight out of Law School' (2020) abs/2010.02559 CoRR <https://arxiv.org/abs/2010.02559>

Chalkidis I and others, 'Neural Contract Element Extraction Revisited' (2021) abs/2101.04355 CoRR <https://arxiv.org/abs/2101.04355>

Fedus W, Zoph B and Shazeer N, 'Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity' (2021) abs/2101.03961 CoRR <https://arxiv.org/abs/2101.03961>

Feldmann Á and others, 'HILBERT, Magyar Nyelvű BERT-Large Modell Tanítása Felhő Környezetben', *XVII. Magyar Számítógépes Nyelvészeti Konferencia* (Szegedi Tudományegyetem TTIK, Informatikai Intézet 2021) <http://real.mtak.hu/id/eprint/120856>

Ferreira TC and others, 'Neural Data-to-Text Generation: A Comparison between Pipeline and End-to-End Architectures' (2019) abs/1908.09022 CoRR <http://arxiv.org/abs/1908.09022>

Indurkhya N and Damerau FJ, *Handbook of Natural Language Processing* (2., Chapman & Hall/CRC 2010)

Jurafsky D and Martin JH, *Speech and Language Processing* (3rd draft, 2020) <https://web.stanford.edu/~jurafsky/slp3/ed3book_dec302020.pdf>

Kilián I, 'Szemantikai Alapú Jogi Tudás-Menedzsment Technológiák' (phdthesis, Pécsi Tudományegyetem Állam- és Jogtudományi Kar, Doktori Iskola 2013) <https://ajk.pte.hu/files/file/doktori-iskola/kilian-imre/kilian-imre-vedes-ertekezes.pdf>

Lauritsen M, ‘Current Frontiers in Legal Drafting Systems’ (2007) <https://static1.squarespace.com/static/571acb59e707ebff3074f461/t/5946f1e39de4bb69d253380c/1497821669156/CurrentFrontiers.pdf>

Markovic M and Gostojic S, ‘Knowledge-Based Legal Document Assembly’ (2020) abs/2009.06611 CoRR <https://arxiv.org/abs/2009.06611>

Nemeskey DM, ‘Introducing HuBERT’, *XVII. Magyar Számítógépes Nyelvészeti Konferencia* (Szegedi Tudományegyetem TTIK, Informatikai Intézet 2021) <http://real.mtak.hu/id/eprint/120856>

Reiter E, ‘NLG vs Templates: Levels of Sophistication in Generating Text’ (September 2016) <https://ehudreiter.com/2016/12/18/nlg-vs-templates/>

Reiter E, ‘Generated Texts Must Be Accurate!’ (September 2019) <https://ehudreiter.com/2019/09/26/generated-texts-must-be-accurate/>

Reiter E, ‘Academic NLG Should Not Fixate on End-to-End Neural’ (December 2020) <https://ehudreiter.com/2020/12/01/dont-fixate-on-end-to-end-neural/>

Reiter E and Dale R, *Building Natural Language Generation Systems* (Cambridge University Press 2000)

Sprowl JA, ‘Automating the Legal Reasoning Process: A Computer That Uses Regulations and Statutes to Draft Legal Documents’ (1979) 4 American Bar Foundation Research Journal 1 <http://www.jstor.org/stable/828045>

Yang ZG, Perlaki A and Laki LJ, ‘Automatikus Összefoglaló Generálás Magyarnyelvre BERT Modellel’ (2020) 16 Magyar Számítógépes Nyelvészeti Konferencia 343 <http://acta.bibl.u-szeged.hu/id/eprint/67658>

Zhang Y and others, ‘When Do You Need Billions of Words of Pretraining Data?’ (2020) abs/2011.04946 CoRR <https://arxiv.org/abs/2011.04946>