Organizing Contexts as a Lattice of Decision Trees for Machine Reading Comprehension

Boris Galitsky[1], Dmitry Ilvovsky[2] and Elizaveta Goncharova[2,3]

[1]*Knowledge Trail Inc, San Jose, CA, USA*

[2]*Higher School of Economics, Moscow, Russia*

[3]*AIRI, Moscow, Russia*

Abstract

Supported decision trees that have been first proposed to boost the performance and the explainability of the expert systems built upon the texts can become a great basis for the machine reading comprehension (MRC) systems. The supported decision tree is based on building and combining the corresponding discourse trees for the text passage. In this work, we build an environment of supported decision trees for the MRC task. Each answer is represented by a path of a supported decision tree and the whole corpus of answers is then form a lattice of supported decision trees. This environment gives a boost to MRC performance, handling cases where it is nontrivial to determine which document/passage MRC needs to be applied to.

Keywords

decision tree, Machine reading comprehension, discourse structure

1. Introduction

Machine reading comprehension (MRC) is a question answering task where the goal of the model is to read and understand text passages and answer the question about them. MRC is designed to check the language model's ability to understand text written in natural language, thus, in some cases, an answer cannot be retrieved from the given text passage, or it requires some world knowledge to answer a question. In such cases, a model should have access to the external knowledge base to retrieve the correct answer. The most promising technique to answer such questions is to augment the language model with the external knowledge database, such as Wikipedia, and to ensemble it with the additional retrieval component that supports the system with the relevant documents [1, 2].

Thus, in the cases, when an answer cannot be retrieved directly from a text, the system is split into two core components [3], where the former is an information retrieval system designed to identify useful pieces of text from the knowledge source (the retriever); and a system to produce the answer given the retrieved documents and the question (the reader).

✉ bgalitsky@hotmail.com (B. Galitsky); dilvovsky@hse.ru (D. Ilvovsky); egoncharova@hse.ru (E. Goncharova)

🄳 0000–0003–0670–8520 (B. Galitsky); 0000–0002–5484–372X (D. Ilvovsky); 0000–0001–8358–9647 (E. Goncharova)

The existing models that perform this problem are based on the transformer architecture and retrieve the relevant text passages based on the constructed latent representations. The main drawback of these techniques is the lack of explainability and limitation of the external documents that a model has access to (e.g., Wikipedia only). In this work, we propose to use supported decision trees DecTSup, first presented in [4] for expert systems as the basis for the relevant passages retrieval. The system allows to enrich the current textual corpora with external documents and use the organized rule-based structure in order to retrieve the relevant text passages that contain an answer to the asked question.

In comparison to our previous work, we show that the DecTSups can be built for the texts of the arbitrary genre. For the experimental evaluation, we refer to the MRC problem in the medical domain and show that utilizing DecTSup-based retrieval procedure improves the performance of the QA model over the standard MRC pipelines by up to 6% for the F1-score.

In a conventional MRC architecture, documents are not organized in any structure, and once a passage deemed most relevant is retrieved, the other documents are ignored. However, when humans answer questions, the answer is backed up by supporting documents so that the answer can be explained. In the real world question answering, the corpus of available documents serves the purpose of providing additional information and clarification on the answer topic. In this work we attempt to reproduce this feature of systematized MRC and organize the documents and passages into such structure as *lattice*. As multiple questions for a set of documents are answered, the involved documents are embedded into this lattice to form chains of explanation for obtained answer. We analyze the advantages of this lattice-based MRC architecture for delivering precise and explainable answers in a systematic way.

2. Background

2.1. Supported Decision Trees

DecTSups have been first presented in [4] as the basis for the expert systems that should retrieve some instructions from the textual data based on the input query. There, the authors claim that a flow of potential recommendations can be easily retrieved from the textual data and organized in the format of the decision tree (DecT) as for the standard numerical attributes. This flow of recommendations or instructions is extracted in the form of a discourse tree [5], where the nodes of the DecT are the elementary discourse units (EDUs) obtained from the discourse tree expressing some condition, and the edges are some type of the rhetorical relations that connect EDUs corresponding to different nodes. The DecT constructed from textual data can be enriched with the additional knowledge retrieved from the specific rhetorical relation and, thus, to construct the *supported decision tree* (DecTSup). DecTSup can be easily integrated into the expert system as a basis to perform dialogue management that improves information retrieval procedure.

To turn a DecT into a corresponding DecTSup, each edge should be labeled with the information extracted from text for the given decision step:

1. The extracted entity;
2. The extracted phrase for the attribute for this entity;

3. The rhetorical relation;

4. The full nucleus and satellite EDUs.

3. Decision Chains and Discourse Structure

3.1. Rhetorical Structure Theory and Decision Chains Construction

A discourse tree (DT) is a basis for the DecT and DecTSup respectively. DT is a hierarchical structure that describes the relations that hold between text units in a document. Several theories have been proposed in the past to describe the discourse structure, among which RST [6] is one of the most developed. During discourse parsing, one can segment a document into non-overlapping text spans (contiguous units for clauses) called EDUs. Each of these EDUs can be tagged as either a nucleus or a satellite, where nucleus nodes are more central and satellite nodes more peripheral. Nucleus units consist of the main information the author expresses in the text, and satellite units contain additional information supporting the one presented in a nucleus. The EDU itself can be of different lengths, i.e., it can contain just one word or a word sequence. The discourse units are organized in the hierarchy by rhetorical relations (e.g., *Antithesis*, *Elaboration*, *List*, etc.) that reflect the function of these EDUs. As a result, we can represent the discourse structure of the text as a DT, where the relations at the top level cover the relations at the bottom.

To build the corresponding DecT, first, a *decision chain* should be retrieved from the DT. A decision chain is defined as a sequence of EDUs with rhetorical relations between sequence elements [7]. Elements of a decision chain are connected with *rhetorical$_r$elation* between a premise and a decision. It can be read as "If ⟨*premise*⟩ then make ⟨*decision*⟩ according to *rhetorical_relation*". In a decision chain, each consecutive member starting from the second one is a ⟨*decision*⟩.

An example of the decision chains and a fragment of DecTSup for a text passage presented below is given in Figure 1.

"Although there is no cure for type 2 diabetes, studies show it is possible for some people to reverse it. Through diet changes and weight loss, you may be able to reach and hold normal blood sugar levels without medication. This does not mean you are completely cured. Type 2 diabetes is an ongoing disease. Even if you are in remission, which means you are not taking medication and your blood sugar levels stay in a healthy range, there is always a chance, that symptoms will return. But it is possible for some people to go years without trouble controlling their glucose and the health concerns that come with diabetes."

elaboration
  explanation
  contrast
    TEXT: Although there is no cure for type 2 diabetes,
    attribution
     TEXT: studies show
     enablement
      TEXT: it is possible for some people
      TEXT: to reverse it.
  evaluation

condition
  TEXT: Through diet changes and weight loss,
  manner-means
    TEXT: you may be able to reach and hold normal blood sugar levels
    TEXT: without medication.
  TEXT: This does not mean you are completely cured.
elaboration
  TEXT: Type 2 diabetes is an ongoing disease.
contrast (RightToLeft)
  elaboration(RightToLeft)
    same-unit
      condition
        TEXT: Even if you are in remission,
        joint
          TEXT: which means you are not taking medication
          TEXT: and your blood sugar levels stay in a healthy range ,
        TEXT: there is always a chance ,
      TEXT: that symptoms will return .
    background
      TEXT: But it is possible for some people to go years
      elaboration
        TEXT: without trouble
        elaboration
          TEXT: controlling their glucose and the health concerns

When a text is represented as a discourse tree, it is split into elementary discourse units (EDUs), denoted by 'TEXT' tag. EDUs are organized hierarchically according to rhetorical relations between them. Using the constructed DT, for an arbitrary rhetorical relation, we can define some patterns defining, how EDUs are connected to each other. In particular, relation of *Elaboration*, ⟨satellite⟩ elaborates (provides additional information) on ⟨nucleus⟩. Certain rhetorical relations have an obvious interpretations in terms of what decision ⟨satellite⟩ can be made by means of ⟨nucleus⟩. For example, *Enablement*(*result*) ⟹ possible to achieve result ⟨nucleus⟩ by the way of ⟨satellite⟨. Based on this logical connection identified by the specific EDUs, we can retrieve possible decision chains for the read passage given bellow:

diet changes and weight loss $\Rightarrow^{condition}$ *without medication* $\Rightarrow^{manner-means}$ *sugar(normal)*
Remission $\Rightarrow^{condition}$ *not taking medications sugar(normal)* $\Rightarrow^{elaboration}$ *chance symptoms(yes)*
OR Remission $\Rightarrow^{contrast}$ *control(sugar(normal))* $\Rightarrow^{elaboration}$ *symptoms(no)*

We denote *sugar(normal)* as a formal representation of target values. Formal representations are shown in italic, and original text – in a regular font.

chance, possibility are the modalities which do not change the configuration of a DecTSup but control the probability of navigation of the given decision chain.

Formally, a decision chain is defined as a sequence of EDUs with rhetorical relations between sequence elements [7]. Each element is a whole original EDU or its representation as a logic form that can be obtained as a result of a semantic parsing: it depends whether an entity from this EDU occurs in an available ontology or not. We encourage the readers to refer to [8] for a
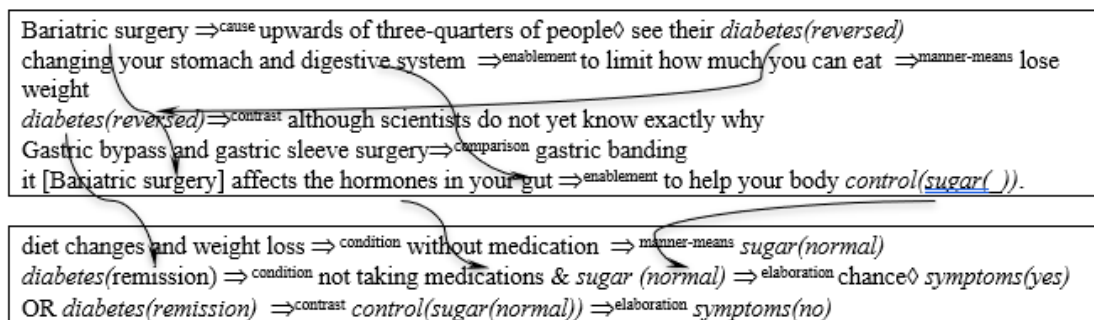
Figure 1: Two sections of decision chains

detailed explanation of how to obtain semantic-based representation of the EDU. For formalized elements of decision chains, it is easier to establish a correspondence or synonymy between entities to form a decision navigation graph.

Elements of a decision chain are connected with $\Rightarrow^{rhetorical\_relation}$ between a premise and a decision. It can be read as "If ⟨premise⟩ then make ⟨decision⟩ according to *rhetorical_relation*". In a decision chain, each consecutive member starting from the second one is a ⟨decision⟩. Each previous member is a ⟨premise⟩.

Figure 1 shows two sections of decision chains extracted from the two texts above. Arrows connect the same (or corresponding) entities, (possibly, parameterized diferently) such as *control*(*sugar*(_)) → *sugar*(*normal*)).

In the first formalized decision expression *control*(*sugar*(_)), the outermost predicate is *control*(_) that ranges over control subjects such as *sugar*(_) with an anonymized variable '_'.

4. DecTSup Environment for Machine Reading Comprehension

4.1. Basic Example

MRC is a task to retrieve the correct answer span from the given textual context. However, the context which is given to MRC system usually restricts its applicability in real-life applications. As, in many cases, the given passage may not contain the necessary information. Efforts made in multi-passage MRC research have somewhat broken the limitation of the given context, but there is still a long way to go as how to ⬚nd the most relevant resources for MRC systems effectively determines the performance of answer prediction. Moreover, the existing DL MRC systems often fail to capture long-range dependencies existing in a text, thus, even if an answer can be retrieved from a context, the system may not be able to find it. It calls for a deeper combination of information retrieval and machine reading comprehension, which we do in this work.

Let us consider the example, where in order to answer a simple question, MRC model should be aware of the additional information.

Passage: There is an ice cube in a glass of water. When the ice cube melts, will the water level have risen, fallen, or remained the same? We have an ice cube floating in the water. If it is floating in equilibrium, then it will have to displace enough water to support its weight. When the ice has melted, it turns into exactly the same volume as it displaced before. So the added volume is the same, so the level of the water will not change.

Question: Into what does ice melt?

Answer: The same volume as it displaced before.

Relying on this text, the transformer-based MRC system cannot answer a very basic question about melting ice into water. Required knowledge is not spelled out in explanation but instead is assumed to be known to the reader. As MRC has no means to acquire it, this knowledge should be added in some way or another. A complete hypothetical decision tree for inferring a solution to a physics problem contains hints on which texts and/or ontology expressions are needed to answer all question about the physical system described in the formulated problem.

We postulate a hypothesis that answering a question, given a passage constitutes navigating a fragment of a decision tree build from this passage. While this approach seems natural when this passage describes some form of a decision explicitly, and certain abstraction is required for an arbitrary text genre where decisions are hypothetical.

4.2. DecTSup MRC Architecture

We propose to treat any text as some kind of problem formulation so that a respective decision tree would tell the MRC system which knowledge is necessary to have complete domain coverage. Usually, a text describes just a single path in a decision tree. Mining for other texts helps reconstruct texts for other paths. Otherwise, it is unclear if even basic questions can be answered (see 4.1). Hence forming a DecTSup for a given passage assure better domain coverage so that the user can expect any relevant question to be answered reasonably well.

If a question requires building a logical chain of facts or forming a decision, the construction of a decision tree in the course of question-answering seems natural. The assumption we make in this study is that a logical chain of facts and respective decision structure is associated with any text, of an arbitrary genre. Considering a given text passage within a decision structure is necessary to determine which other passages need to be involved. This approach is expected to be much more robust and precise than a traditional information retrieval (IR) based, where candidate passages are determined based on keywords.

Under regular MRC architecture, an IR component first finds a document and a passage, and then MRC finds an exact answer in this passage. In the proposed MRC architecture, the IR system finds candidate documents along with DecTSup built for these documents, if available. Then the DecTSup-MRC components use the lattice of DecTSups to decide on which passages need to be involved in the answer.

We draw the feature-by-feature comparison of three MRC environments: default, retrieval-augmented [9] and DecTSup-based in Table 1.

In Figure 2, we present the architecture for the standard MRC pipeline and MRC with the DecTSup.

Table 1
Comparison of the IR-based MRC environments

Feature / Architecture	Default MRC	Retrieval-augmented approach	DecTSup-based
How passages and answers are organized	Unordered set of passages and answers	A model decides which passages to involve	Passages are linked with each other via DecTSups which are in turns organized in a lattice
Treatment of texts describing decisions	–	No special treatment	Decision structure is reflected in how passages are organized. Relying on an assumption that a question concern a single decision step
Inter-connection between answers / passages / document	–	implicit	Answers are interconnected
How links between passages and documents can be expressed	–	implicit	Via common parts of a path in a DecTSup
Explanation for why the exact answer is chosen	–	Implicit, via optimization	Via the full path in DecTSup
Explanation for why the documents containing the answer are selected	–	Implicit, via optimization	Via the lattice of DecTSups
Explanation for why certain entities occur in the answer	–	–	–
Training for document retriever	–	marginalization over sets of retrieved documents is approximated using an expectation-maximization algorithm.	–
Providing background info	–	–	A search session can be followed by a navigation session

4.3. An Algorithm for Building DecTSup Environment

The essence of building DecTSup environment for MRC is organizing mapping between set of passages P and DecTSups. This algorithm is given a set of documents/passages P_s in a corpus and a set of external documents/passages E_s, and also given a sequence of queries Q_s against P_s, builds a set of supported decision trees DecTSups.

An answer a is obtained as a function MRC($\langle q, p \rangle$). A true answer a_{true} may need another passage, not necessarily p, including other passages p' external document h belong to E_s. The
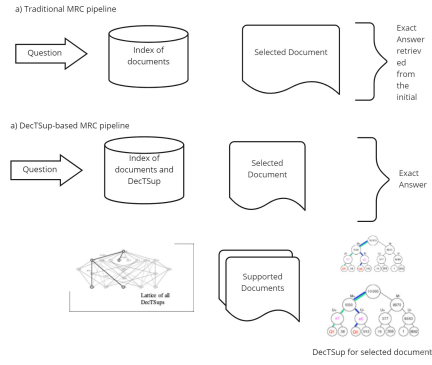
Figure 2: MRC pipelines: a) classical NRC pipeline; b) retrieval-based MRC pipeline based on DecTSup

whole passage p corresponds to a DecTSup(p) and an answer a – to a path of this DecTSup$(p) \leftarrow path_{DecTSup(p)}(a)$. There is a many-to-many mapping between multiple passages and DecTSups.

An algorithm for building this DecTSup is as follows.

Initially, DecTSup $= \emptyset$. Given a corpus of passages P_s, iterate through all available queries Q_s and build a set $\cup_{p \in P} DecTSup(p)$.

For each $\langle q, ? \rangle$

1. Compute $a = MRC(\langle q, p \rangle)$.
2. Compute a_{true} checking if passage p is suitable. If not find another p' or a suitable external doc $p = h$, using the lattice L. Decide whether to substitute or to augment p.
3. Build a chain of phrases from a_{true} for DecTSup(a).
4. Build $path_{DecTSup(a)}$ from this chain of phrases. Only $path_{DecTSup(p)}$ is currently available, not the whole tree.
5. Extend $path_{DecTSup(a)}$ towards a DecTSup by turning all phrases into negations.
6. Update the DecTSups system, identifying a location for $path_{DecTSup(a)}$:
a. Find existing path $path_{DecTSup(p)}$ for $\langle q, p \rangle$ and verify $path_{DecTSup(p)}$ covers $path_{DecTSup(a)}$.
b. If not, find current DecTSup(p) and augment it with $path_{DecTSup(a)}$. Check for overlapping nodes and do the path merge if appropriate;
c. If no such DecTSup(p) exists, form a fragment of new DecTSup(p) from $path_{DecTSup(a)}$.

Once all available queries $q \in Q_s$ are ran, organize a set of DecTSup(p) into a lattice (see 4.4). When a new batch of queries arrive, re-apply this algorithm and then recompute the lattice.

The procedure of extension of $path_{DecTSup(a)}$ towards a $DecTSup$ is implemented by turning all phrases into negations works as follows. For each phrase associated with a path node, we consider its negation and branch it off this node. If a phrase in $path_{DecTSup(a)}$ is already a negation, remove this negation.

This algorithm is also naturally applied when P is a book split into passages, the result can be viewed as a book hyperlink graph.

During MRC environment construction, each query produces an answer that maps into a path in the DecTSup- based MRC environment. Initially, most queries form new paths which
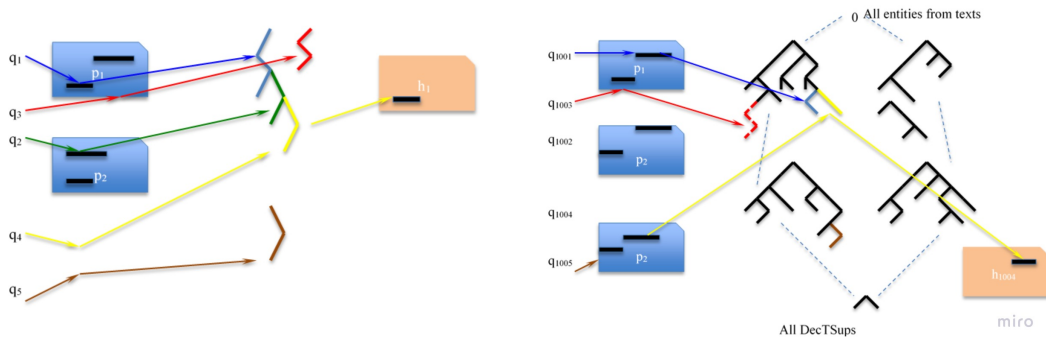
Figure 3: Building the lattice of DecTSup.

are then merged into a part of the future DecTSups. Once new batches of queries does not construct new DecTSups, we form a lattice from them.

In Figure 3, we show the end of the learning session for building the lattice of DecTSups. Most individual DecTSups are now built and new queries do not initiate additions of new paths to DecTSups. Most of the DecTSups are completed and now form a lattice (shown by dotted lines). Black nodes of DecTSup show nodes that have been formed by the time queries q1001, ... are launched.

Parts of the DecTSups corresponding to the given queries are shown in color. For example, query q1001 is connected by the blue arrow with its answer and then with its blue path in the top-left DecTSup. This blue path has just been added. On the contrary, query q1002 is mapped into the existing part of the DecTSup (shown in dotted red). Query q1001 requires a new external passage h1004.

4.4. Decision Trees and Concept Lattices

The following phase is responsible for retrieving the relevant text passages from the obtained DecTSupp of the documents. In this work, we propose to form a concept lattice of all DecTSups for a corpus of documents to enable this corpus with a structure assuring effective MRC. For a text and its DecTSup, we extract entities and their attributes used for decisions and form the formal context.

We use the essential ideas of Lindig [10] algorithm which efficiently generates formal concepts together with their subconcept-superconcept hierarchy in the form of concept lattice. The algorithm builds the concept lattice by iteratively generating the neighbor concepts of a concept $\langle A, B \rangle$, either top-down the lattice by adding new attributes to concept intents or bottom-up by adding new objects to concept extents.

For each $A \subseteq X$ and $B \subseteq Y$ we denote by A' a subset of Y and by B' a subset of X defined by
$A' = \{y \in Y | \text{ for each } x \in A : hx, yi \in I\}$,
$B' = \{x \in X | \text{ for each } y \in B : hx, yi \in I\}$.
That is, A' is the set of all attributes (text entities) from Y shared by all objects (DecTSups)

from A (and similarly for B').

A formal concept consists of a set A (so-called extent) of objects which fall under the concept and a set B (so-called intent) of attributes that fall under the concept such that A is the set of all objects sharing all attributes from B and, conversely, B is the collection of all attributes from Y shared by all objects from A.

The algorithm is based on the fact that a concept $\langle C, D \rangle$ is a neighbor of a given concept $\langle A, B \rangle$ if D is generated by $B \cup \{y\}$, i.e. $D = (B \cup \{y\})''$, where $y \in Y \backslash B$ is an attribute such that for all attributes $z \in D$ –- B it holds that $B \cup \{z\}$ generates the same concept $\langle C, D \rangle$, i.e. neighbors of $\langle A, B \rangle = \{\langle C, D \rangle | D = (B \cup \{y\})'', y \in Y \backslash B$ such that $(B \cup \{z\})'' = D$ for all $z \in DB\}$.

Then a selection of a tree of concepts from the part of the concept lattice occurs. First, for each concept $c = \langle A, B \rangle$ the number L_c of all of its lower concepts is computed. Each lower concept is counted for each different attribute added to the concept c, l_c. For instance, if a concept $d = \langle C, D \rangle$ is generated from concept c by adding either attribute x or attribute y (i.e. $D = (B \cup \{x\})''$ or $D = (B \cup \{y\})''$, respectively), the concept d is counted twice and l_c is increased by two.

Then a tree of concepts is chosen from the part of the concept lattice by iteratively going from the greatest concept (generated by no attributes or, equivalently, by all objects) to minimal concepts. The selection is based on the number l_c of lower concepts of the currently considered concept c.

4.5. Online Selection of Passages for a Query

At search time, to select the passage, we match the query with a chain of phrases for a path in a DecTSup. We try to find such DecTSup so that as many chain phrases path(i) match query phrases as possible.

$$DecTSup : \sum_i score(q \wedge path(i)_{DecTSup}) \to max$$

$\wedge$ is a syntactic generalization operator [11].

Once a single DecTSup is identified, we select the passages p_s associated with matched nodes of DecTSup.

In some cases, the best match between the query and path phrases can be distributed through multiple DecTSups. Then the condition of connectedness in the lattice L of DecTSups must be maintained:

$$DecTSups(j) : \sum_{i,j} score(q \wedge path(i)DecTSup(j)) \to max \ \&$$

$$\& \ node(DecTSups(j) \in DecTSups(k)) \in L.$$

To summarize the preference for the occurrence of an answer in a corpus of documents, we prefer it to be in a single passage. If it is not possible, we extend this passage towards other passages connected vis a DecTSup. If it is still insufficient for answer identification, we further extend passages towards foreign DecTSups linked in a lattice. Finally, in the worst case scenario, passages come from unrelated documents associated with DecTSups anywhere in the lattice.

Table 2
Evaluation results

Search domain	Baseline F1 (IR-based)	F1 with synt. gen. between q and p	F1 with a set of DecTSups	F1 with a lattice of DecTSups
Bloating	76.3	77.0	79.3	80.4
Cough	75.2	76.8	76.9	82.4
Diarrhea	77.0	75.9	78.0	79.3
Dizziness	77.9	77.1	79.8	81.9
Fatigue	72.6	75.4	80.4	82.7
Fever	71.9	74.5	78.0	79.3
Headache	74.6	74.8	81.5	82.7
Muscle Cramp	77.3	76.0	80.6	81.0
Nausea	72.1	76.4	78.4	80.7
Throat irritation	75.9	73.2	79.5	82.3
Average	75.1	75.7	79.2	81.3

Linked lattice nodes correspond to passages containing the same entities plus minus one or two, therefore, closely related. This is maintained by how the lattice is defined on the set of all DecTSups.

5. Evaluation

We check the MRC model performance on the collection of medical instruction collected from the WebMD website and syntactically generate the question that could be asked regarding the retrieved passages. We select ten classes of diseases to diversify the experiments, and track each processing step to identify the performance bottleneck. It should be mentioned that the F1-score reported to assess the model's performance is applied for each topic independently assessing whether a retrieved answer is correct or not.

In the baseline, we rely on IR to identify passages in the fixed corpus of documents, so the main source of errors is an improperly selected passage or a lack of appropriate passage in the corpus. Our second baseline in the third column is syntactic generalization between query q and passage p which is better than keyword frequency maintained by IR but lags behind the approach developed in this paper. In the fourth column, we show F1 of DecTSup-enabled MRC environment. The fifth column shows the contribution of ideal, corrected DecTSups and the last, sixth column should the contribution of the lattice of DecTSups, where individual trees are organized to systematically identify additional passages which need to be involved in finding the exact answer.

The second baseline is only 0.6% better then the IR-based document identification, which makes the syntactic generalization insufficient for the robust identification of relevant passages to form the exact comprehensive answer. We observe that an unstructured MRC can have a boost of 4% F1-score by finding the best document by DecTSup environment. Proceeding from IR passage selection to the one based on independent DecTSups turns out to be an ultimate win

for the MRC environment. A further upgrade from a set of DecTSups to a lattice gives further 2% boost in search performance.

6. Conclusion

In this work, we explored a way to build decision trees from text relying on discourse analysis and use this environment to boost the MRC model's performance. We use DecTSup environment for retrieving the relevant documents and passages, where the answer can be found. We compare the DecTSup-based retrieval procedures with several baselines including syntactic generalization for matching questions and passages and keywords matching on the set of the medical instructions texts. We show that utilizing the DecTSup as the description of texts and combining them into the concept lattice improves the performance of MRC by up to 6% on average. In future studies, we will consider building a concept lattice from a textual description of data [11], instead of a decision tree for authors' instructions on how to do things and make decisions in the course of it.

References

[1] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, W. Cohen, Open domain question answering using early fusion of knowledge bases and text, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 4231–4242. doi:10.18653/v1/D18-1455.

[2] Y. Wang, K. Liu, J. Liu, W. He, Y. Lyu, H. Wu, S. Li, H. Wang, Multi-passage machine reading comprehension with cross-passage answer verification, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 1918–1927. doi:10.18653/v1/P18-1178.

[3] D. Chen, A. Fisch, J. Weston, A. Bordes, Reading Wikipedia to answer open-domain questions, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 1870–1879. doi:10.18653/v1/P17-1171.

[4] B. Galitsky, Chapter 3 - Obtaining supported decision trees from text for health system applications, Academic Press, 2022. doi:https://doi.org/10.1016/B978-0-12-824521-7.00013-2.

[5] B. Galitsky, Matching parse thickets for open domain question answering, Data Knowledge Engineering 107 (2017) 24–50. doi:https://doi.org/10.1016/j.datak.2016.11.002.

[6] W. Mann, S. Thompson, Rhetorical structure theory: Toward a functional theory of text organization, Text 8 (1988). doi:10.1515/text.1.1988.8.3.243.

[7] B. Galitsky, Managing Customer Relations in an Explainable Way, Springer International Publishing, Cham, 2020, pp. 309–377. doi:10.1007/978-3-030-52167-7_8.

[8] B. A. Galitsky, D. Ilvovsky, Validating correctness of textual explanation with complete discourse trees, in: FCA4AI@IJCAI, 2019.

[9] D. S. Sachan, S. Reddy, W. Hamilton, C. Dyer, D. Yogatama, End-to-end training of multi-document reader and retriever for open-domain question answering, in: NeurIPS, 2021.

[10] C. Lindig, Fast concept analysis, in: Working with Conceptual Structures – Contributions to ICCS 2000, Shaker Verlag, 2000, pp. 152–161.

[11] B. Galitsky, G. Dobrocsi, J. Rosa, S. Kuznetsov, Using generalization of syntactic parse trees for taxonomy capture on the web, volume 6828, 2011, pp. 104–117. doi:10.1007/978-3-642-22688-5_8.