Knowledge-Based Systems 193 (2020) 105422

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A deep-learning approach to mining conditions☆

Fernando O. Gallego *, Rafael Corchuelo

Universidad de Sevilla, ETSI Informática. Avda. de la Reina Mercedes, s/n. Sevilla E-41012, Spain

ARTICLE INFO

Article history:
Received 26 November 2018
Received in revised form 20 December 2019
Accepted 22 December 2019
Available online 31 December 2019

Keywords:
Natural language processing
Text mining
Condition mining
Neural networks

ABSTRACT

A condition is a constraint that determines when a consequent holds. Mining them in text is paramount to understand many sentences properly. In the literature, there are a few pattern-based proposals that fall short regarding recall because it is not easy to characterise unusual ways to express conditions with hand-crafted patterns; there is one machine-learning proposal that is bound to the Japanese language, requires specific-purpose dictionaries, taxonomies, and heuristics, works on opinion sentences only, and was evaluated very shallowly. In this article, we present a deep-learning proposal to mine conditions that does not have any of the previous drawbacks; furthermore, we have performed a comprehensive experimental study on a large multi-lingual dataset on many common topics; our conclusion is that our proposals are similar to the state of the art in terms of precision, but improve recall enough to beat them in terms of F_1 score.

1. Introduction

The Web is an immense information repository. It is then not surprising that there is a growing interest in processing web documents to extract the data that they provide. Text miners process such documents and extract structured information that can be used to feed business processes. Common text miners include entity-relation extractors, opinion miners, and recommenders. Unfortunately, they do not deal well with conditional sentences because they neglect the conditions and focus on the consequents, which may easily lead to wrong conclusions.

Entity-relation extractors mine entities and relations, which results in overviews that are useful to make decisions. Next, we provide some examples that clearly motivate the need for mining conditions and make it clear how important they are regarding making correct decisions. For instance, current systems [1,2] return a relation of the form ("NBT Bank", "won't merge", "Alliance Bank") from a sentence like "may the new law be approved and NBT Bank will not merge Alliance Bank"[1]; neglecting condition "may the new law be approved" might lead a broker not to recommend investing on NBT Bank regardless of the status of the new law. Opinion miners analyse the opinions that people express in social media, which is very useful for companies to improve their products or to devise marketing campaigns. Current systems [3,4] return an assessment of the form {"flash" = −0.99, "lens" = 0.55} from a sentence like "the flash's horrible when used indoors, but the lens is good enough for amateurs"; neglecting conditions "when used indoors" and "for amateurs" might result in a manufacturer not testing the flash appropriately or targeting the wrong customers. Recommenders suggest items building on a user's profile that may include purchase records as well as chat messages. Current systems [5] typically start placing advertisements about rent-a-car agents and flights when they see a sentence like "I will rent a Chevy later, when I book my flight", independently from whether condition "when I book my flight" holds or not.

The previous examples clearly motivate the need for mining conditions, which has been seldom explored in the literature. The naivest proposals search for user-defined patterns that build on syntactic anchors [6,7]. They generally attain good precision, but fall short regarding recall because the distribution of connectives that are used to introduce conditions is long-tail; that is, there are a few usual ways to introduce conditions that can be easily modelled using user-defined patterns, but too many unusual ways that cannot be easily modelled using such an approach. There is only one machine-learning proposal [8], which was intended to recognise as many patterns as possible in the long-tail distribution. Unfortunately, it is bound to the Japanese language, it must be customised with several specific-purpose dictionaries, taxonomies, and heuristics, mines conditions regarding opinions only, and it was evaluated on a small dataset with 3155 sentences that were sampled from hotel reviews.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105422.

* Corresponding author.
E-mail addresses: fogallego@us.es (F.O. Gallego), corchu@us.es (R. Corchuelo).

[1] The sample sentences come from actual reviews, which means that some of them are grammatically incorrect.

https://doi.org/10.1016/j.knosys.2019.105422
0950-7051/

In this article, we present a proposal to mine conditions. It has been developed in close co-operation with Opileak.es, which is a commercial text-mining service from which we have learnt the importance of mining conditions so as not to misinterpret the results of sentiment analysers. It relies on a deep-learning regression approach to rank a set of candidates that are computed from the dependency tree of the input sentence; such trees have been widely used in several text mining tasks, e.g., machine translation [9,10], summarisation [11,12], and semantic role labelling [13,14]. This proposal clearly improves on the state of the art. The theoretical improvements are as follows: it does not require to provide any user-defined patterns and it relies on a deep neural learning approach that transforms the input sentences into a rich feature space in which it is relatively easy to learn a predictor that can identify the tokens that belong to a condition. The practical improvements are as follows: it can mine both factual and opinion sentences and it does not require any specific-purpose resources. The superiority of our approach has been experimentally proven on a dataset with 4.7M sentences on 16 common topics in 4 mainstream languages, where we have beaten the few proposals in the literature in terms of F_1 score.

The rest of the article is organised as follows: Section 2 reports on the related work; Section 3 introduces some preliminary concepts; Section 4 describes our proposal; Section 5 reports on our experimental analysis; and, finally, Section 6 presents our conclusions.

2. Related work

Narayanan et al. [15] range amongst the first authors who realised the problem with conditions in the field of opinion mining. They devised a feature model that helps machine-learn a regressor that computes the polarity of a conditional sentence, but they did not report on a proposal to mine their conditions; they assumed that the sentences were pre-classified as either conditional or non-conditional sentences by a person. Recently, Skeppstedt et al. [16] presented a complementary proposal that can automatically classify a sentence in such categories, but neither was it their goal to mine their conditions.

The naivest proposals to mine conditions are based on searching for user-defined patterns that rely on syntactic anchors. Mausam et al. [6] studied the problem in the field of entity-relation extraction; they realised that many usual conditions can be identified by locating adverbial clauses whose first word is one of the sixteen one-word condition connectives in English; unfortunately, they did not report on the effectiveness of their proposal to mine conditions, only on the overall effectiveness of their proposal for entity-relation extraction. Their system was updated recently with new features [17], but their proposal to mine conditions was not. Chikersal et al. [7] reported on a similar, but simpler proposal: search for sequences of words in between connectives "if", "unless", "until", and "in case" and the first occurrence of word "then" or a comma. Unfortunately, the previous proposals are not appealing because there are many unusual ways to introduce conditions, which makes hand-crafting patterns with high recall very difficult; furthermore, it is not straightforward to adapt them to other languages in which common connectives are multi-word or there is not a unique, non-contextual translation for some English connectives. We confirmed the previous claims with our experimental analysis.

The only existing machine-learning proposal was introduced by Nakayama and Fujii [8], who worked in the field of opinion mining in Japanese. They devised a model that is based on features that are computed by means of a syntactic parser and a semantic analyser. The former identifies so-called bunsetus, which are Japanese syntactic units that consists of one independent word and one or more ancillary words, as well as their inter-dependencies; the latter identifies opinion expressions, which requires to provide some specific-purpose dictionaries, taxonomies, and heuristics. They used Conditional Random Fields and Support Vector Machines to learn classifiers that make bunsetus that can be considered conditions apart from the others. Unfortunately, their proposal was only evaluated on a small dataset with 3155 sentences from hotel reviews and the best F_1 score attained was 0.58. As a conclusion, this proposal is not generally applicable and its effectiveness seems to be poor.

Our conclusion is that some researchers have already realised that mining conditions is of uttermost importance for text miners not to return information that can be easily misinterpreted in cases in which conditional sentences are involved. Unfortunately, the problem remains quite unexplored since the few existing proposals have many drawbacks that hinder their general applicability. This motivated us to work on a new proposal that overcomes their weaknesses and outperforms them.

3. Preliminaries

In this section, we introduce some preliminary concepts that are required to describe our proposal.

Preliminary 1 (*Conditionals*)**.** A conditional sentence, or conditional for short, is composed of two clauses, namely: a condition and a consequent. The condition describes a state, a factor, or a circumstance that must hold so that the consequent holds.

Preliminary 2 (*Usual Conditionals*)**.** Usual conditionals are expressed by means of grammatical patterns that rely on connectives and verb tenses that are well-known in the literature. The patterns make a difference amongst zero-conditionals, which convey general truths, first conditionals, which convey possible conditions and their likely results, second conditionals, which convey hypothetical conditions and their likely results, and third conditionals, which convey unreal past conditions and their likely results in the past.

Example 1. For instance, "if you heat ice, then it melts" is a zero conditional sentence; "Unless I do not pass my exams, I will get my degree" is a first conditional sentence; "I had quit my job as long as I were not in debt" is a second conditional sentence; and "we had've attended the show even if she had not let us know" is a third conditional sentence.

Preliminary 3 (*Unusual Conditionals*)**.** Unusual conditionals do not fit the patterns that characterise the previous types of conditionals. There is not a standard set of connectives or verb tenses to introduce their conditions.

Example 2. For instance, in sentence "put a grain in the tank and the engine will break", the condition "put a grain in the tank" describes an action that is sufficient to break an engine. In sentence "people who like The Beatles will enjoy The Kinks", condition "who like The Beatles" expresses a taste that characterises a group of people that will likely enjoy "The Kinks". In sentence "after you book Aladdin, you will get a discount for a DVD", condition "after you book Aladdin" expresses a specific time after which one can get a discount. In sentence "it smells terribly bad near the main gate", the condition "near the main gate" expresses a place where it does not smell well. In sentence "you must show your ID due to law restrictions", the condition "due to law restrictions" expresses the reason why an ID is mandatory.

```
method train(ds) returns r
  T := ∅
  for each (s, L) ∈ ds do
    C := generateCandidates(s)
    for each c ∈ C do
      z := computeScore(c, L)
      T := T ∪ {(c, z)}
    end
  end
  r := learnRegressor(T)
end
```

```
method apply(s, r, θ) returns R
  C := generateCandidates(s)
  R := ∅
  for each c ∈ C do
    z := apply r to c
    if z > θ then
      R := R ∪ {(c, z)}
    end
  end
  R := removeOverlaps(R)
end
```

Fig. 1. Main methods of our proposal.

Preliminary 4 (*Deep Learning*)**.** Deep learning revolves around a number of machine-learning methods whose focus is on learning feature-based data representations of the input data that facilitate learning classifiers or regressors. They typically build on non-linear transformations that are organised in layers so that the outputs of a layer constitute the inputs of the succeeding one. Many deep learning approaches build on neural networks. They have achieved relevant results in computer vision [18,19] and natural language processing (NLP) [20,21]. In the case of NLP, it is necessary to transform the input text into vectors using so-called word embedders [22].

Preliminary 5 (*Deep Neural Networks*)**.** A deep neural network is an artificial neural network with multiple hidden layers. There are two models that are very appropriate in NLP, namely: recurrent neural networks (RNN) and convolutional neural networks (CNN). An RNN is a neural network in which the connections between its units form a directed graph across a sequence [23], which makes them particularly well-suited to deal with sequences of data in which each element depends on the previous ones. Bi-directional recurrent neural networks (BiRNN) [24] are a particular class of recurrent neural networks that can take both the past and the future elements of a sequence into account. Unfortunately, both RNNs and BiRNNs suffer from the so-called exploding and vanishing gradient problems [25,26], which can be addressed by controlling the data that is passed on to the next training epoch by means of gated recurrent units (GRUs) [23] or bi-directional gated recurrent units (BiGRUs) [27]. A CNN [28,29] is a class of neural network that is composed of convolution layers, which consist of computation units whose purpose is to transform some regions of the input vectors by means of non-linear functions, and pooling layers, which consists of filters that are intended to subsample the output of the previous convolutions in an attempt to reduce the number of parameters that need to be computed in the network. Finally, it is worth mentioning other kinds of neural networks, including Boltzmann Machines (BM) [30], Restricted Boltzmann Machines (RBM) [31], and Extreme Learning Machines (ELM) [32]. BMs and RBMs rely on a type of stochastic recurrent neural network and Markov random fields; ELMs are feed-forward neural networks in which the parameters of the hidden nodes do not need to be tuned, and consequently, they do not require back-propagation.

4. Our proposal

In this section, we describe our proposal to mine conditions, which consists in learning a regressor that assesses how likely a candidate condition is an actual condition. Hereinafter, we refer to condition candidates as candidates and to actual conditions as conditions since there is no room for confusion. We first describe the main methods and then provide an insight into the ancillary methods.

4.1. Main methods

Fig. 1 shows the main methods of our proposal, namely: *train*, which learns a regressor that assesses candidates, and *apply*, which computes the best candidates in a sentence.

Method *train* takes a dataset *ds* as input and returns a regressor r. The input dataset is of the form $\{(s^{(i)}, L^{(i)})\}_{i=1}^{n}$, where each $s^{(i)}$ denotes a sentence and each $L^{(i)}$ denotes a set of labels that identify the conditions in that sentence ($n \geq 0$). The output regressor is a function that given a candidate returns a score that assesses how likely it is an actual condition. The method first initialises training set T to the empty set and then loops over dataset *ds*; for each sentence s and set of labels L in *ds*, it first computes a set of candidates; then, for each candidate c, it computes a score z and stores a tuple of the form (c, z) in training set T. When the main loop finishes, it learns a regressor from T using a deep-learning approach.

Method *apply* takes a sentence s, a regressor r, and a threshold θ as input and returns a set R of tuples of the form $\{(c^{(i)}, z^{(i)})\}_{i=1}^{n}$, where each $c^{(i)}$ denotes a candidate and $z^{(i)}$ its corresponding score, which must be equal to or greater than threshold θ ($n \geq 0$). The method first generates the candidates in s, stores them in set C, and initialises R to an empty set; it then iterates over set C; for each candidate c in set C, it first computes its score by applying regressor r to it; if it is equal to or greater than threshold θ, then candidate c is added to the result set. When the main loop finishes, R provides a collection of candidates and scores; before returning it, we must remove the candidates that overlap others with a higher score. The candidates in R are considered the actual conditions in the input sentence s.

4.2. Generating candidates

Our first ancillary method is *generateCandidates*, which takes a sentence as input and returns a set of candidates. A naive approach would simply generate as many sub-strings as possible, but it would be very inefficient because a sentence with n words has $O(n^2)$ such sub-strings. In order to reduce the candidate space, we use a sequence of non-overlapping blocks that are computed from a dependency tree.

Method *generateCandidates* first computes the dependency tree of the input sentence [33], changes the words in its nodes to lowercase, and stems them. It then computes a sequence of non-overlapping blocks, which are sequences of tokens of the form (w, d, p), where w denotes a stem, d the dependency tag that links its node in the dependency tree to its parent, if any, and p its position in the sentence. To compute a non-overlapped sequence of blocks we first recursively select the sub-trees whose depth is exactly two because it helps to select syntactical units like noun-phrases, adjective-phrases, or verb-phrases; then, we repeat the procedure for the nodes whose depth is equal to one

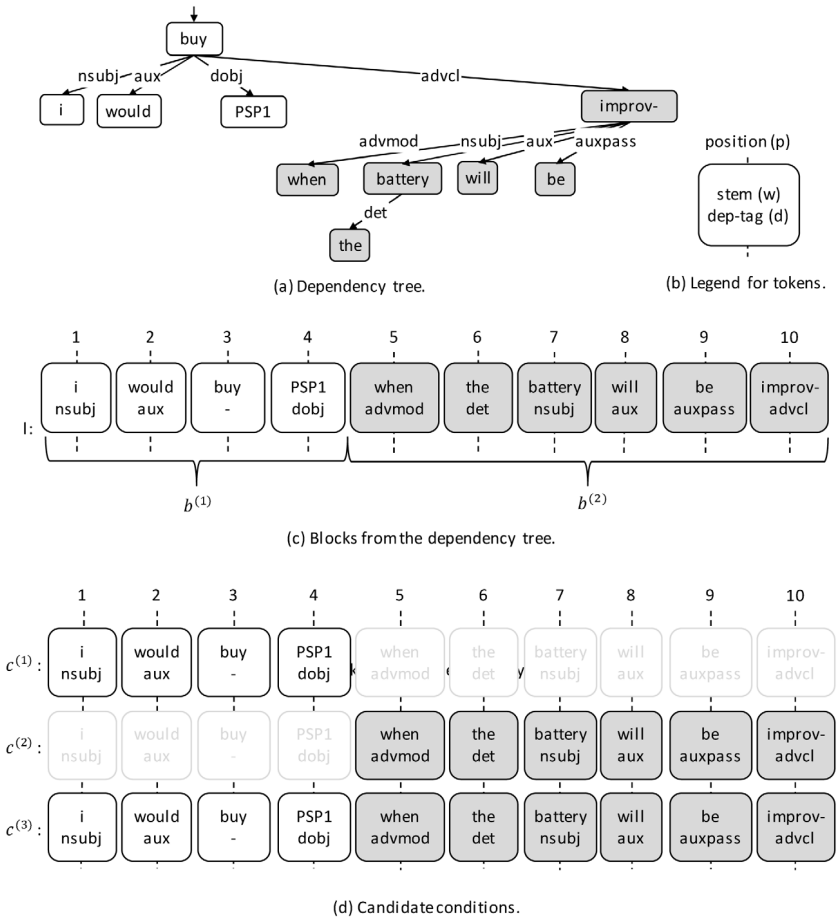

Fig. 2. Sample candidate generation.

because it helps to select smaller syntactical units; and, finally, we select the nodes that have only one token. The depth of a sub-tree is the distance from its root node to its deepest leaf node. Note that we need to ensure that the tokens in a block are consecutive. For that reason, we split every block that contains a hole. Finally, we generate the candidates as the sequences of tokens from all of the sequential combinations of the blocks.

Example 3. Fig. 2a shows the dependency tree of sentence "I would buy PSP1 when the battery will be improved". The nodes that correspond to the condition are highlighted in grey. Fig. 2b shows a legend that helps understand how a token (w, d, p) is mapped onto our graphical notation. Fig. 2c shows the blocks that are generated from the previous dependency tree. Fig. 2d shows the candidates that are generated from the previous blocks. Candidate $c^{(1)}$ is generated from block $b^{(1)}$, candidate $c^{(2)}$ is generated from block $b^{(2)}$, and candidate $c^{(3)}$, which corresponds to the whole sentence, is generated from blocks $b^{(1)}$ and $b^{(2)}$.

4.3. Computing matching scores

Our second ancillary method is *computeScore*, which takes a candidate c and a set of labels L as input and returns its corresponding score. A naive approach would simply return 0.00 if c does not exactly match any of the labels in L and 1.00 otherwise, but that is too crisp. An approach in which a candidate gets a score in range [0.00, 1.00] captures the chances that it is an actual condition much better.

Our idea was to use an approach that is based on the well-known F_1 score in order to balance the precision and the recall of a candidate. The F_1 score is computed as $\frac{2\,tp}{(tp+fp)+(tp+fn)}$, where tp, fp, and fn denote, respectively, the number of true positives, false positives, and false negatives. Given a candidate c and a label l, it makes sense to interpret the tokens that they have in common as true positive tokens, the tokens in c that are not in l as false positive tokens, and the tokens in l that are not in c as false negative tokens. We also realised that the first few tokens in a condition typically provide an anchor that characterises it by means of a connective. (In Section 5, we analyse them and their influence on conditions.) So, we decided to measure the matching between a candidate and a label as follows:

$$match(c, l) = \sum_{i=1}^{|l|} \begin{Bmatrix} 1/i & \text{if } l_i \in c \\ 0 & \text{otherwise} \end{Bmatrix}$$

Simply put: let l_i denote the ith token in the label ($i = 1..|l|$); if l_i is in the candidate, we then add $1/i$ to the score and zero otherwise. This way, the first few tokens in the label contribute much more to the score than the remaining ones. That is, given a candidate c and a label l, $match(c, l)$ may be interpreted as a measure of the number of true positive tokens in c.

Given the previous definition, the maximum matching for a candidate or a label x is defined as follows:

$$match^*(x) = \sum_{i=1}^{|x|} 1/i$$

Realise that given a candidate c, $match^*(c)$ is a measure of the number of true positive tokens (the tokens that belong to both the candidate and the label) plus the false positive tokens

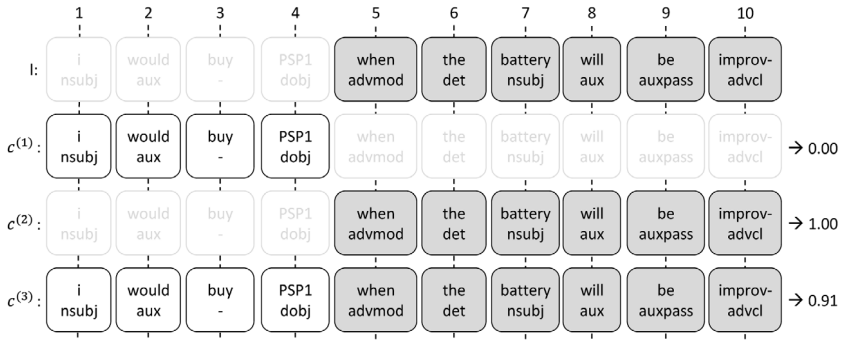

Fig. 3. Sample matching scores.

(the tokens that belong to the candidate, but not to the label); similarly, given a label l, $match^*(l)$ is a measure of the number of true positive tokens (the tokens that belong to both the label and the candidate) plus the number of false negative tokens (the tokens that belong to the label, but not to the candidate).

Our proposal to compute the matching score of candidate c with respect to the set of labels L is then as follows:

$$score(c, L) = \max_{l \in L} \frac{2\ match(c, l)}{match^*(c) + match^*(l)}$$

Note the similarity to the F_1 score since $match(c, l)$, $match^*(c)$, and $match^*(l)$ are measures of tp, $tp+fp$, and $tp+fn$, respectively.

Example 4. Fig. 3 shows label l, which corresponds to the condition in our running example, and how the candidates match it. Candidate $c^{(1)}$ does not match any true positive token, but four false positive tokens and six false negative tokens, which results in a matching score of 0.00. Candidate $c^{(2)}$ matches the label perfectly, i.e., it matches six true positive tokens and no false positive or false negative token, which results in a matching score of 1.00. Candidate $c^{(3)}$ represents the whole input sentence, which obviously contains the label, i.e., six true positive tokens, but also four false positive tokens, which results in a matching score of 0.91.

4.4. Learning a regressor

Our third ancillary method is *learnRegressor*, which takes a training set T as input and returns a regressor r.

Prior to learning a regressor, the candidates in the training set must be vectorised; note that it is necessary to put a limit to the maximum candidate length, which is referred to as η, and that padding must be used when analysing shorter candidates. Given a candidate of the form $\langle (w_i, d_i) \rangle_{i=1}^{n}$, we transform it into a sequence of the form $\langle \overline{w_i} \oplus \overline{d_i} \rangle_{i=1}^{\eta}$, where $\overline{w_i}$ denotes the vectorisation of stem w using a word embedder [22], $\overline{d}$ denotes the vectorisation of dependency tag d using one-hot encoding [34], $\overline{w} \oplus \overline{d}$ the vector that results from catenating the previous ones, and η denotes the size of the longest possible condition; padding tokens need to be added if the original condition is shorter than η tokens. Note that the vectorisation of a condition can then be interpreted as a matrix with η rows and δ columns, where δ denotes the dimensionality of the word embedding vectorisation plus the dimensionality of the one-hot vectorisation.

Fig. 4 summarises some of the neural network architectures that we have devised. We used the Mean Squared Error [35] as the loss function and we trained them using Stochastic Gradient Descent Optimisation [36] with batch size 32. In order to prevent over-fitting as much as possible, we used some drop-out regularisations [37] and early stopping [38] when the loss did not improve significatively after 10 epochs. We did not apply a decay momentum because we observed that the loss always converges smoothly, even if it needs more epochs in some cases.

Our baseline architectures are the following: a shallow multi-layer perceptron (MLP1), which has an input layer and an output layer, and a deep multi-layer perceptron (MLP2), which has an input layer, a hidden layer, and an output layer. We devised a dozen more architectures with several combinations of MLP, GRU, BiGRU, and CNN layers and different configurations of activation functions, loss functions, regularisation parameters, and optimisation methods. In the sequel, we focus on the best architectures, namely: gated recurrent units (GRU), bi-directional gated recurrent units (BiGRU), convolutional neural networks (CNN), and a hybrid approach that combines convolutional neural networks and bi-directional gated recurrent units (CNNBiGRU). The CNN network includes two convolution layers and a pooling layer. Our proposal is to use a convolution layer with a large number of filters in order to create a wide range of first-level features, but a smaller number of filters in the second convolution layer to obtain a more specific range of second-level features that combine the first ones. Finally, the pooling layer combines the previous deep features using a global maximum function as the global pooling strategy since our experiments prove that it performs very well. The CNNBiGRU network uses a convolution with a number of filters similar to the input length, and then applies a local pooling that captures the most relevant features only. We then apply a BiGRU layer that takes the dependencies between tokens into account, from both the beginning to the end of the sentences and vice versa.

4.5. Removing overlaps

The fourth ancillary method is *removeOverlaps*, which takes a set of tuples of the form (c, z) as input, where c denotes a candidate and z its corresponding score, and removes the tuples whose candidates overlap a candidate with a higher score. In other words, given the input set of tuples R, it computes the following subset:

$$\{(c, z) \in R \mid \nexists (c', z') : (c', z') \in R \wedge c' \cap c \neq \emptyset \wedge z' > z\}$$

We do not provide any additional details since this method can be implemented very straightforwardly.

Example 5. Assume that the threshold to select the best candidates is set to $\theta = 0.50$. In our running example, method *apply* would return candidates $c^{(2)}$ and $c^{(3)}$ since they are the only whose scores exceed the threshold, cf. Fig. 3. Note that both candidates overlap, so the one with the lowest score is filtered out. In this case, method *apply* would then return condition $c^{(2)}$ only, which, indeed, represents the condition in our running example.

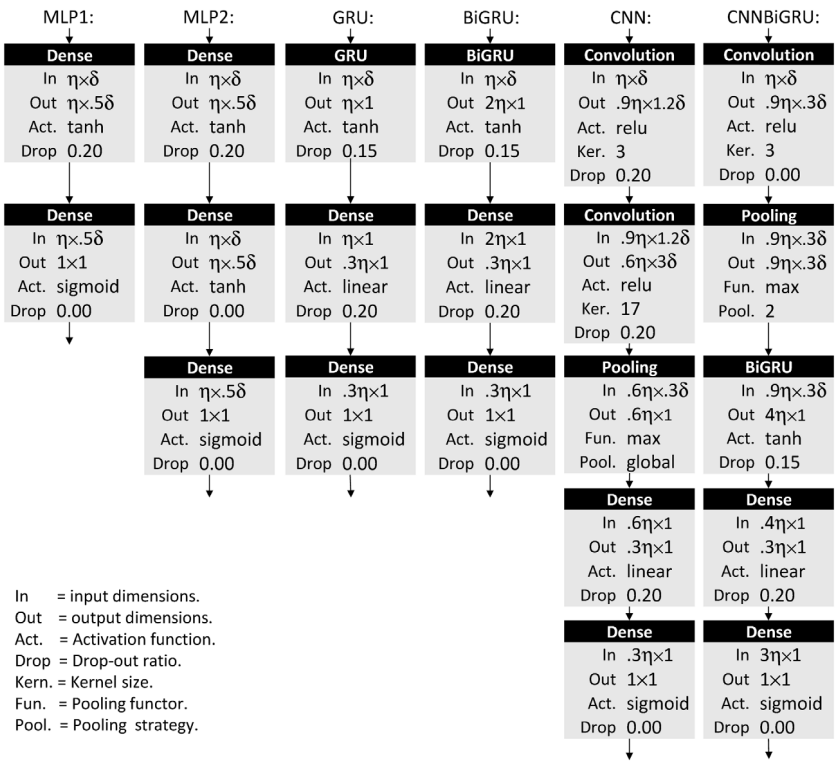

Fig. 4. Neural network architectures.

5. Experimental analysis

In this section, we first describe our computing machinery, the dataset used for evaluation, the baselines, and the alternatives compared; next, we report on our experimental results; finally, we present our statistical analysis.

5.1. Computing machinery

We implemented our proposal[2] using Python 3.5.4, Snowball 1.2.1 to compute word stems, the Stanford NLP Core Library 3.8.0 to generate dependency trees, Gensim 2.3.0 to compute word embedders using a Word2Vec implementation, and Keras 2.0.8 with Theano 1.0.0 for training our neural networks. We run our experiments on a virtual computer that was equipped with one Intel Xeon E5-2690 core at 2.60 GHz, 2 GiB of RAM, and an Nvidia Tesla K10 GPU accelerator with 2 GK-104 GPUs at 745 MHz with 3.5 GiB of RAM each; the operating system on top of which we run our experiments was CentOS Linux 7.3.0.

5.2. Our dataset

In this subsection, we first describe our dataset of conditions,[3] then analyse how conditional connectives are distributed, and, finally, analyse how similar conditions are.

Description. We have not found any public datasets with conditions, which motivated us to create one. It consists of 4 671 533 sentences in English, Spanish, French, and Italian that were gathered from Ciao.com between April 2017 and May 2017. The sentences were classified into 16 topics according to their sources, namely: adults, baby care, beauty, books, cameras, computers, films, headsets, hotels, music, ovens, pets, phones, TV sets, and video games. For each sentence, we labelled its conditions and made it explicit where their connectives start and end.

To create the word embedding efficiently without degrading effectiveness, we replaced numbers, e-mail addresses, URLs, and words whose frequency was equal or smaller than five by "NUMBER", "EMAIL", "URL", and "UNK", respectively.

In Table 1, we provide a summary of our dataset. The columns of the table denote the language (Lang), the domain (Domain), the number of conditions found (#Conds), the number of sentences (#Sents), the number of sentences that we have labelled as of the time of writing this article (#Lab), the number of sentences that contain at least one condition (#SwC), and the corresponding percentage (%SwC).

In Fig. 5a and b, we present a box and whisker plot that represents the number of words per condition and sentence in our dataset. Regarding our candidate ranking proposal, we set the length of the candidates to $\eta = 50$ and the length of the input sentences to $\lambda = 100$ since these thresholds are enough for the vast majority of sentences in our dataset. Note that these thresholds do not miss any conditions, but some extremely long sentences that can be considered outliers.

Connective distribution. In Table 2, we show the frequency of two groups of connectives, namely: the five most frequent and the five around the 75th percentile. In every language, there are a few usual connectives that have high frequencies and many other connectives with low frequencies.

The previous figures suggest that the distribution of connectives is a long-tail distribution. To confirm this intuition, it is necessary to compare the distribution of connectives to the Power-Law and Log-Normal distributions, which are the standard long-tail distributions, and to the Exponential distribution, which is not long-tail by definition [39,40].

In Fig. 6, we plot the Complementary Cumulative Distribution Functions (CCDF) of the previous distributions. It is not difficult to

[2] The implementation is available at https://github.com/FernanOrtega/candidate-ranking.

[3] The dataset is available at https://www.kaggle.com/fogallego/reviews-with-conditions.

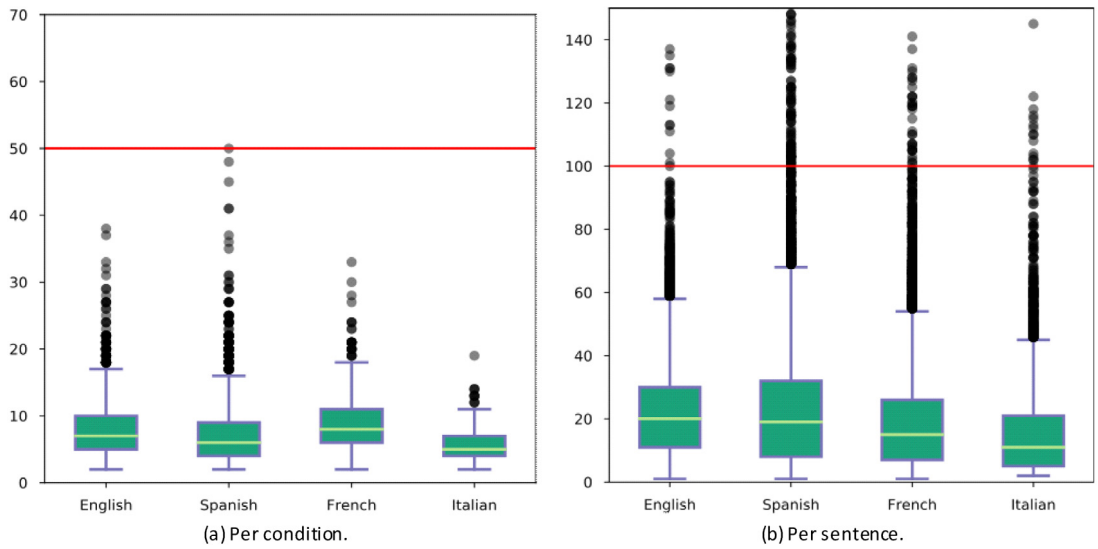

Fig. 5. Typical numbers of words.

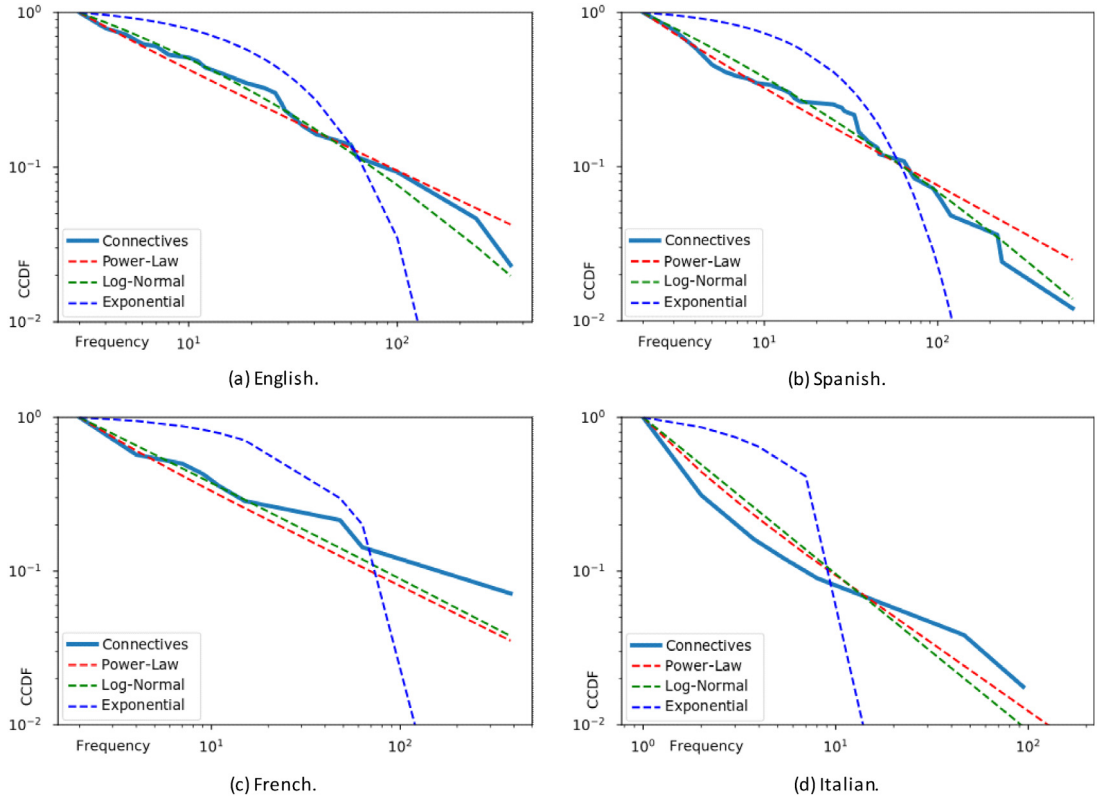

Fig. 6. Connective distribution.

realise that the Log-Normal distribution and the Power-Law distributions are very similar to the connective distribution, whereas the Exponential one is not. We conducted a Likelihood Ratio Test [41] to check the previous idea statistically. The results of the test are shown in Table 3: for each language in our dataset, we compared the distribution of connectives to every two pairs of the previous standard distributions and computed R as the log likelihood ratio and the corresponding p-value. Independently from the language, the comparison to the Power-Law distribution and the Log-Normal distribution returns p-values that are greater than the standard significance level ($\alpha = 0.05$), which indicates that there is not enough empirical evidence to prove that the connective distribution is significantly different from a Power-Law or a Log-Normal distribution; note that the comparisons to the Power-Law and the Exponential distributions or the Log-Normal and the Power-Law distribution return a positive log likelihood ratio with a p-value that is smaller than the standard significance level, which indicates that there is enough empirical evidence to prove that the connective distribution is similar to the Power-Law or the Log-Normal distributions, but different from the Exponential distribution.

The conclusion is that there is enough statistical evidence to consider the connective distribution a long-tail distribution. Simply put, relying on a collection of handcrafted patterns will

Table 1
Summary of our dataset.

Lang	Domain	#Conds	#Sents	#Lab	#SwC	%SwC
en	Adults	325	2 164	1007	290	28.80%
	Babycare	26	28 282	1008	22	2.18%
	Beauty	19	109 348	1007	18	1.79%
	Books	3	49 653	1002	3	0.30%
	Cameras	10	8 574	1010	10	0.99%
	Films	3	45 547	1024	3	0.29%
	Hotels	231	21 908	1002	214	21.36%
	Music	113	24 218	1000	106	10.60%
	Pets	284	34 878	1002	251	25.05%
	Phones	172	2 052	1027	162	15.77%
	Tvsets	175	3 537	1007	159	15.79%
	Video games	176	111 439	1006	159	15.81%
es	Books	597	1 013 746	2005	502	25.04%
	Computers	359	102 021	1006	303	30.12%
	Films	210	209 810	1004	185	18.43%
	Headsets	243	10 222	1022	195	19.08%
	Hotels	173	317 177	1028	162	15.76%
	Music	114	417 456	1033	100	9.68%
	Oven	177	17 197	1011	152	15.03%
	Pets	130	66 845	1017	114	11.21%
	Phones	146	547 921	1010	132	13.07%
	Tvsets	132	108 287	1005	108	10.75%
	Video games	158	537 921	1008	140	13.89%
fr	Adults	42	7 586	1012	42	4.15%
	Babycare	16	47 972	1023	16	1.56%
	Beauty	32	74 739	1004	30	2.99%
	Books	42	94 141	1002	40	3.99%
	Cameras	61	3 641	1008	54	5.36%
	Computers	59	2 792	1006	56	5.57%
	Films	59	136 068	1010	54	5.35%
	Hotels	46	12 941	1000	46	4.60%
	Music	23	25 109	1004	21	2.09%
	Pets	49	13 991	1003	45	4.49%
	Phones	62	7 972	1001	61	6.09%
	Tvsets	39	6 410	1009	37	3.67%
	Video games	66	77 457	1007	61	6.06%
it	Adults	40	5 068	464	40	8.62%
	Books	5	5 890	554	5	0.90%
	Computers	2	3 403	430	2	0.47%
	Films	2	267 454	1236	2	0.16%
	Headsets	139	4 703	1020	129	12.65%
	Hotels	37	35 915	895	35	3.91%
	Music	77	46 029	1170	68	5.81%
	Oven	48	2 049	1041	44	4.23%

typically fall short in terms of recall because there are too many ways to introduce conditions, which clearly argues for a machine-learning solution.

Condition similarity. We have also analysed the similarity of the conditions in our dataset. Our goal was to check if there are groups of conditions that are similar enough to be modelled using some common features, e.g., verbs, adverbs, or prepositions. To carry this analysis out, we changed every word into lowercase and then computed a vectorisation of each condition as follows: each component of the vectors corresponds to a different word and represents its tf-idf frequency in the condition being vectorised [42]. The vectorisations have 2311 words in English, 3796 words in Spanish, 1386 words in French, and 658 words in Italian.

To visualise them, we performed dimensionality reduction by means of Isomap [43]. Furthermore, we computed the Gaussian Kernel Density Estimation [44] to better visualise the density of samples with Scott's Rule [45] to compute the estimator bandwidth.

In Fig. 7, we show a graphical representation of the Isomap projections of conditions. The hues range from bright yellow, which represents the highest densities (conditions that are very similar to each other), to dark blue, which represents the lowest densities (conditions that are not similar to each other). It is not difficult to realise that the conditions are organised as follows: there is one small group with high density, a larger group with average density, and a very large group with low density.

As a conclusion, it must not be difficult for a person to learn a rule to mine instances of the first group since there are many examples available and they seem very similar to each other; but it must not be that easy to deal with the many other conditions since they are not similar to each other. This also argues for a machine-learning solution.

Table 3
Fitting the connective distribution.

Lang	$Dist_1$	$Dist_2$	R	*p*-value
en	Power-Law	Log-Normal	−1.14	0.33
	Power-Law	Exponential	19.32	0.04
	Log-Normal	Exponential	20.47	0.01
es	Power-Law	Log-Normal	−0.77	0.31
	Power-Law	Exponential	64.32	0.00
	Log-Normal	Exponential	65.09	0.00
fr	Power-Law	Log-Normal	−0.06	0.46
	Power-Law	Exponential	15.40	0.01
	Log-Normal	Exponential	15.46	0.01
it	Power-Law	Log-Normal	−7.00	0.11
	Power-Law	Exponential	70.02	0.00
	Log-Normal	Exponential	77.02	0.00

5.3. Baselines and alternatives

We used the proposals by Chikersal et al. [7] and Mausam et al. [6] as baselines. The proposal by Nakayama and Fujii [8] was not

Table 2
Connectives samples.

(a.1) Top.			(a.2) 75th percentile.			(b.1) Top.			(b.2) 75th percentile.		
Lang	Connective	Freq	Lang	Connective	Freq	Lang	Connective	Freq	Lang	Connective	Freq
en	if	349	en	every time	2	es	si	600	es	a medida que	3
	when	239		over	2		cuando	235		además de	3
	for	100		regardless of	2		para	221		porque si	3
	after	100		regardless	2		en	119		incluso cuando	3
	before	64		prior	2		al	105		a causa de	3
	while	60		but in	2		a	94		viendo	3
(c.1) Top.			(c.2) 75th percentile.			(d.1) Top.			(d.2) 75th percentile.		
Lang	Connective	Freq	Lang	Connective	Freq	Lang	Connective	Freq	Lang	Connective	Freq
fr	si	384	fr	sauf si	2	it	se	170	it0	in	2
	même si	63		avec	2		quando	108		in cerca	2
	pour	48		même s′	2		per	7		da	2
	mais si	15		si bien qu′	1		anche	7		soprattutto	2
	en cas de	11		si on ne	1		in cui	4		ogni volta	2
	quand	9		quant	1		da quando	3		qualora	1

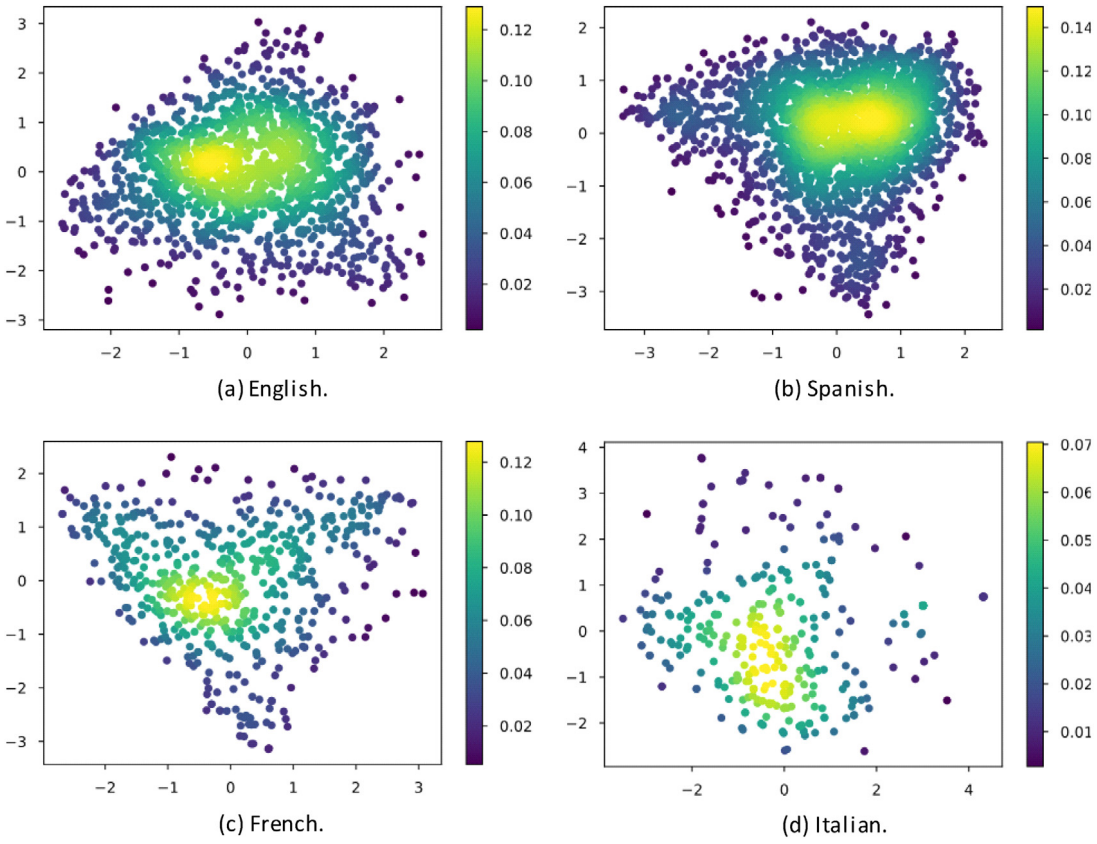

Fig. 7. Similarity of conditions.

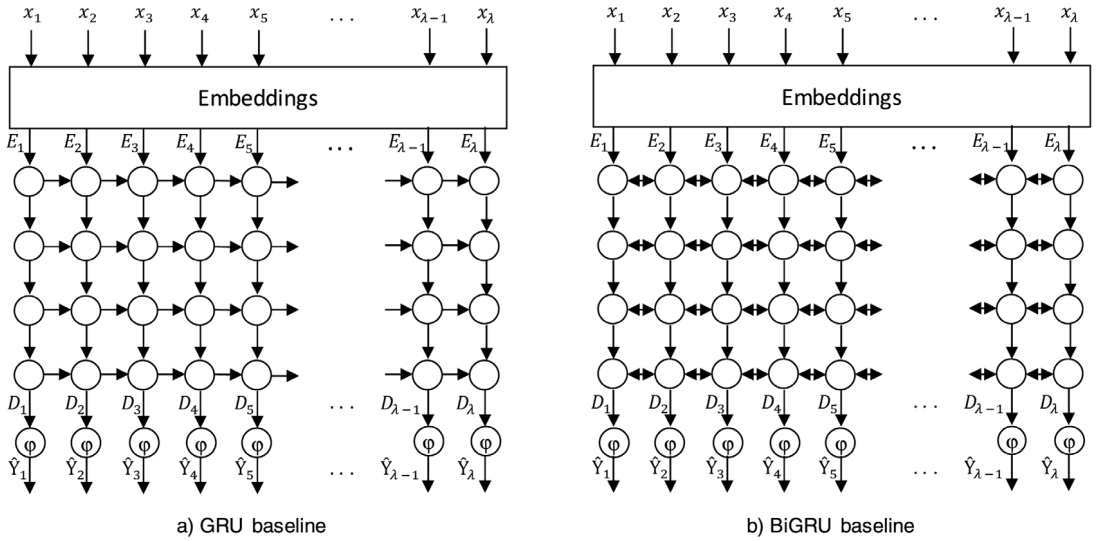

Fig. 8. GRU baseline.

taken into account because it is bound to the Japanese language and it is not clear how it can be extended to deal with other languages.

Fig. 8 represents two additional approaches that we used as baselines, namely: a recurrent neural network with GRU units (GRU) and a recurrent neural network with BiGRU units (BiGRU). Both approaches use word embeddings to transform the input sentence $(x_1, x_2, \ldots, x_\lambda)$ into vectors (E_i) that represent its words. Due to our dataset analysis, we set parameter λ to 100, which is large enough, and padding was used when vectorising shorter sentences. The output is computed from the last recurrent layer as $\widehat{Y}_i = \varphi(W D_i + b)$, where φ is the Sigmoid function, W is a weight matrix, D_i is the output of the decoder, and b is a bias vector. It represents the IOB tags of each word. Each IOB tag determines whether the corresponding word belongs to a condition or not. Then, it is easy to reconstruct the conditions of the sentence. In both cases, we used Categorical Cross Entropy [46] as the loss function, drop-out regularisations [37] to prevent over-fitting, early stopping [38] when the loss did not improve significantly after 10 epochs, the Adam method [47] with batch size 32 as the optimiser, and the sigmoid activation function.

Table 4
Experimental results.

Proposal	English			Spanish			French			Italian		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Mausam et al.	0.63	0.61	0.62	0.67	0.53	0.59	0.65	0.67	0.66	0.49	0.50	0.49
Chikersal et al.	0.80	0.46	0.59	0.80	0.44	0.57	0.88	0.58	0.70	0.50	0.48	0.49
GRU	0.65	0.62	0.63	0.61	0.56	0.59	0.87	0.57	0.69	0.58	0.59	0.59
BiGRU	0.67	0.62	0.64	0.63	0.59	0.61	0.82	0.77	0.80	0.60	0.57	0.58

(a) Baselines.

Alternatives	English			Spanish			French			Italian		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
$MLP1_{0.25}$	0.51	0.80	0.62	0.48	0.81	0.61	0.66	0.87	0.75	0.57	0.75	0.65
$MLP1_{0.50}$	0.58	0.69	0.63	0.56	0.66	0.60	0.71	0.85	0.78	0.63	0.64	0.63
$MLP1_{0.75}$	0.60	0.58	0.59	0.59	0.53	0.56	0.73	0.76	0.74	0.65	0.54	0.59
$MLP2_{0.25}$	0.49	0.82	0.61	0.44	0.84	0.58	0.69	0.87	0.77	0.58	0.80	0.67
$MLP2_{0.50}$	0.56	0.74	0.63	0.52	0.75	0.62	0.75	0.86	0.80	0.64	0.68	0.66
$MLP2_{0.75}$	0.61	0.64	0.63	0.59	0.65	0.61	0.77	0.83	0.80	0.72	0.59	0.65
$GRU_{0.25}$	0.55	0.80	0.65	0.53	0.82	0.64	0.74	0.88	0.80	0.69	0.55	0.62
$GRU_{0.50}$	0.58	0.74	0.65	0.57	0.77	0.66	0.81	0.78	0.79	0.74	0.55	0.63
$GRU_{0.75}$	0.61	0.66	0.63	0.63	0.67	0.65	0.75	0.85	0.80	1.00	0.48	0.65
$BiGRU_{0.25}$	0.55	0.80	0.65	0.53	0.82	0.64	0.74	0.88	0.80	0.93	0.48	0.63
$BiGRU_{0.50}$	0.59	0.74	0.65	0.59	0.76	0.66	0.80	0.67	0.73	0.87	0.48	0.62
$BiGRU_{0.75}$	0.62	0.68	0.65	0.63	0.66	0.64	0.76	0.84	0.79	1.00	0.48	0.65
$CNN_{0.25}$	0.62	0.68	0.65	0.54	0.78	0.64	1.00	0.48	0.65	1.00	0.48	0.65
$CNN_{0.50}$	0.92	0.49	0.64	0.65	0.68	0.67	1.00	0.48	0.65	1.00	0.48	0.65
$CNN_{0.75}$	1.00	0.44	0.61	0.71	0.58	0.64	1.00	0.48	0.65	1.00	0.48	0.65
$CNNBiGRU_{0.25}$	0.55	0.78	0.65	0.53	0.78	0.64	0.76	0.88	0.81	0.82	0.62	0.71
$CNNBiGRU_{0.50}$	0.59	0.73	0.65	0.60	0.73	0.66	0.77	0.88	0.82	0.83	0.58	0.68
$CNNBiGRU_{0.75}$	0.62	0.67	0.64	0.65	0.65	0.65	0.78	0.87	0.82	0.77	0.64	0.70

(b) Our alternatives.

Regarding our proposal, we evaluated our six alternatives using the following values for the threshold: $\theta = 0.25$, $\theta = 0.50$, or $\theta = 0.75$. For the sake of readability, we refer to them using their names and the threshold as subscripts, namely: $MLP1_\theta$, $MLP2_\theta$, GRU_θ, $BiGRU_\theta$, CNN_θ, and $CNNBiGRU_\theta$.

5.4. Experimental results

We evaluated the baselines and our alternatives on our dataset using 4-fold cross-validation. We measured the standard performance measures, namely: precision, recall, and the F_1 score. Table 4 presents the experimental results.

The conclusion from our results is that the state-of-the-art baselines can attain relatively good precision; Mausam et al.'s [6] proposal attains a recall that is similar to its precision, but Chikersal et al.'s [7] proposal falls short regarding recall. The baselines perform slightly worse than our alternatives in most situations. Most of our alternatives beat the baselines regarding recall because they learn complex patterns that a person cannot easily spot. Note that the improvement regarding recall is enough for the F_1 score to improve all of the baselines.

5.5. Statistical analysis

To make a decision regarding which of the alternatives performs the best according to their F_1 score, we used a stratified strategy that builds on Hommel's statistical comparison test at the standard significance level ($\alpha = 0.05$). The test computes the empirical ranking of every alternative; it then compares the best-ranked one to the others by computing a z statistic and its corresponding p-value, which must be compared to the significance level α as follows: if the it is smaller than or equal to α, then the conclusion is that there is enough evidence in the experimental data to support the hypothesis that the difference between two alternatives is statistically significant; otherwise, the conclusion is that the experimental data cannot sustain that there is a statistically significant difference.

In Table 5a, b, and c, we report on the results of the statistical analysis regarding our alternatives. The best-ranked alternative is CNNBiGRU in every case, independently from the value of the threshold. Note that the difference in F_1 score is not significant with respect to the BiGRU, the GRU, or the CNN alternatives when $\theta = 0.25$, it is not significant with respect to the GRU, the BiGRU, the MLP2, or the CNN alternatives when $\theta = 0.50$, and it is not significant with respect to the BiGRU or the GRU alternatives when $\theta = 0.75$. Our conclusion is that the CNNBiGRU alternative is the best one in terms of F_1 score. In Table 5d, we report on the results of comparing the CNNBiGRU alternative with the three values of θ. Note that the p-value is always greater than the standard significance level, which means that the differences in F_1 score are not significant. Thus, we select the CNNBiGRU alternative with $\theta = 0.50$ as the best alternative.

In Table 6, we report on the results of comparing our best alternative to the baselines. The best-ranked alternative is the CNNBiGRU alternative with $\theta = 0.50$. The difference with our recurrent neural network baseline regarding the F_1 score is not significant from a statistical point of view since the p-value is greater than the standard significance level. Note, however, that the difference is very significant with respect to the baselines since the p-value is nearly zero. Thus, we select the CNNBiGRU alternative with $\theta = 0.50$ as our best proposal to mine conditions.

6. Conclusions

In this article, we have motivated the need for mining conditions and we have presented a proposal to address the problem. It relies on a deep-learning regression approach to rank a set of

Table 5
Comparison of our alternatives.

(a) Comparison with $\theta = 0.25$.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.25}$	2.38	–	–
$BiGRU_{0.25}$	2.75	0.57	1.00
$GRU_{0.25}$	3.13	1.13	1.00
$CNN_{0.25}$	3.69	1.98	0.33
$MLP2_{0.25}$	4.31	2.93	0.03
$MLP1_{0.25}$	4.75	3.59	0.00

(b) Comparison with $\theta = 0.50$.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.50}$	2.56	–	–
$GRU_{0.50}$	2.88	0.47	1.00
$BiGRU_{0.50}$	3.25	1.04	1.00
$MLP2_{0.50}$	3.56	1.51	0.78
$CNN_{0.50}$	3.81	1.89	0.59
$MLP1_{0.50}$	4.94	3.59	0.00

(c) Comparison with $\theta = 0.75$.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.75}$	1.56	–	–
$BiGRU_{0.75}$	2.72	1.75	0.26
$GRU_{0.75}$	2.78	1.84	0.26
$MLP2_{0.75}$	3.81	3.40	0.00
$CNN_{0.75}$	4.38	4.25	0.00
$MLP1_{0.75}$	5.75	6.33	0.00

(d) Comparison of best alternatives.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.50}$	1.75	–	–
$CNNBiGRU_{0.75}$	1.88	0.35	0.72
$CNNBiGRU_{0.25}$	2.38	1.77	0.23

Table 6
Comparison to the baselines.

Proposal	Rank	z	p-value
$CNNBiGRU_{0.50}$	1.19	–	–
GRU	2.16	1.98	0.13
BiGRU	2.21	2.05	0.11
Mausam et al.	3.06	4.11	0.00
Chikersal et al.	3.69	5.48	0.00

candidates that are computed from the dependency tree of the input sentence. It does not rely on any user-defined patterns, it does not require any specific-purpose dictionaries, taxonomies or heuristics, it can mine conditions in both factual and opinion sentences, and it is not bound to a specific language. We have also performed an exhaustive experimental analysis on a publicly-available multi-lingual dataset with a large number of sentences on many common topics. Our results confirm that our proposal is similar to the state-of-the-art proposals in terms of precision, but it improves recall enough to beat them in terms of F_1 score. We have sustained the previous conclusions using sound statistical tests. We foresee two interesting future work paths, namely: to explore conditions in question answering systems and to explore transfer learning to adapt our proposal to other languages.

CRediT authorship contribution statement

Fernando O. Gallego: Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Rafael Corchuelo:** Conceptualization, Formal analysis, Data curation, Writing - review & editing, Supervision.

Acknowledgements

Supported by Opileak.es, Spain and the Spanish R&D programme (grants TIN2013-40848-R and TIN2016-75394-R). The computing facilities were provided by the Andalusian Scientific Computing Centre (CICA), Spain. We also thank Dr. Francisco Herrera for his hints on statistical analyses and sharing his software with us.

References

[1] O. Etzioni, A. Fader, J. Christensen, S. Soderland, Mausam, Open information extraction: the second generation, in: IJCAI, 2011, pp. 3–10.
[2] T.M. Mitchell, W.W. Cohen, E.R. Hruschka, P.P. Talukdar, J. Betteridge, A. Carlson, B.D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E.A. Platanios, A. Ritter, M. Samadi, B. Settles, R.C. Wang, D.T. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, in: AAAI, 2015, pp. 2302–2310.
[3] K. Ravi, V. Ravi, A survey on opinion mining and sentiment analysis: tasks, approaches and applications, Knowl.-Based Syst. 89 (2015) 14–46.
[4] K. Schouten, F. Frasincar, Survey on aspect-level sentiment analysis, IEEE Trans. Knowl. Data Eng. 28 (3) (2016) 813–830.
[5] W.X. Zhao, S. Li, Y. He, E.Y. Chang, J. Wen, X. Li, Connecting social media to e-commerce, IEEE Trans. Knowl. Data Eng. 28 (5) (2016) 1147–1159.
[6] Mausam, M. Schmitz, S. Soderland, R. Bart, O. Etzioni, Open language learning for information extraction, in: EMNLP-CoNLL, 2012, pp. 523–534.
[7] P. Chikersal, S. Poria, E. Cambria, A.F. Gelbukh, C.E. Siong, Modelling public sentiment in Twitter, in: CICLing (2), 2015, pp. 49–65.
[8] Y. Nakayama, A. Fujii, Extracting condition-opinion relations toward fine-grained opinion mining, in: EMNLP, 2015, pp. 622–631.
[9] A. Tamura, T. Watanabe, E. Sumita, H. Takamura, M. Okumura, Part-of-speech induction in dependency trees for statistical machine translation, in: ACL, 2013, pp. 841–851.
[10] C. Ding, Y. Arase, Dependency tree abstraction for long-distance reordering in statistical machine translation, in: EACL, 2014, pp. 424–433.
[11] Y. Kikuchi, T. Hirao, H. Takamura, M. Okumura, M. Nagata, Single document summarization based on nested tree tructure, in: ACL, 2014, pp. 315–320.
[12] S.B. Özates, A. Özgür, D.R. Radev, Sentence similarity based on dependency tree kernels for multi-document summarization, in: LREC, 2016.
[13] K. Hacioglu, Semantic role labeling using dependency trees, in: COLING, 2004.
[14] C. Chen, A. Palmer, C. Sporleder, Enhancing active learning for semantic role labeling via compressed dependency trees, in: IJCNLP, 2011, pp. 183–191.
[15] R. Narayanan, B. Liu, A.N. Choudhary, Sentiment analysis of conditional sentences, in: EMNLP, 2009, pp. 180–189.
[16] M. Skeppstedt, T. Schamp-Bjerede, M. Sahlgren, C. Paradis, A. Kerren, Detecting speculations, contrasts, and conditionals in consumer reviews, in: WASSA@EMNLP, 2015, pp. 162–168.
[17] Mausam, Open information extraction systems and downstream applications, in: IJCAI, 2016, pp. 4074–4077.
[18] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-V4, Inception-ResNet, and the impact of residual connections on learning, in: AAAI, 2017, pp. 4278–4284.
[19] P. Tang, H. Wang, S. Kwong, GoogleNet based multi-stage feature fusion of deep CNN for scene recognition, Neurocomputing 225 (2017) 188–197.
[20] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: NIPS, 2014, pp. 3104–3112.
[21] F. Zhai, S. Potdar, B. Xiang, B. Zhou, Neural models for sequence chunking, in: AAAI, 2017, pp. 3365–3371.
[22] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: NIPS, 2013, pp. 3111–3119.
[23] H. Han, S. Zhang, J. Qiao, An adaptive growing and pruning algorithm for designing recurrent neural networks, Neurocomputing 242 (2017) 51–62.
[24] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[25] Y. Bengio, P.Y. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.
[26] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: ICML, 2013, pp. 1310–1318.
[27] M. Nußbaum-Thom, J. Cui, B. Ramabhadran, V. Goel, Acoustic modeling using bi-directional gated recurrent convolutional units, in: Interspeech, 2016, pp. 390–394.
[28] Y. Kim, Convolutional neural networks for sentence classification, in: EMNLP, 2014, pp. 1746–1751.
[29] C.N. dos Santos, B. Xiang, B. Zhou, Classifying relations by ranking with convolutional neural networks, in: ACL (1), 2015, pp. 626–634.
[30] J. Zhang, S. Ding, N. Zhang, Y. Xue, Weight uncertainty in Boltzmann machine, Cogn. Comput. 8 (6) (2016) 1064–1073.
[31] N. Zhang, S. Ding, J. Zhang, Y. Xue, An overview on restricted Boltzmann machines, Neurocomputing 275 (2018) 1186–1199.
[32] N. Zhang, S. Ding, J. Zhang, Multi layer ELM-RBF for multi-label learning, Appl. Soft Comput. 43 (2016) 535–545.
[33] D. Chen, C.D. Manning, A fast and accurate dependency parser using neural networks, in: EMNLP, 2014, pp. 740–750.
[34] X. Zhang, J.J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: NIPS, 2015, pp. 649–657.
[35] A.Y. Aravkin, J.V. Burke, A. Chiuso, G. Pillonetto, Convex versus non-convex estimators for regression and sparse estimation: the mean squared error properties of ARD and GLasso, J. Mach. Learn. Res. 15 (1) (2014) 217–252.
[36] K. Cohen, A. Nedic, R. Srikant, On projected stochastic gradient descent algorithm with weighted averaging for least squares regression, in: ICASSP, 2016, pp. 2314–2318.
[37] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from over-fitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[38] R. Caruana, S. Lawrence, C.L. Giles, Overfitting in neural nets: back-propagation, conjugate gradient, and early stopping, in: NIPS, 2000, pp. 402–408.
[39] F. Chierichetti, R. Kumar, B. Pang, On the power laws of language: word frequency distributions, in: SIGIR, 2017, pp. 385–394.
[40] S. Singh, Y.M. Tripathi, Bayesian estimation and prediction for a hybrid censored Log-Normal distribution, IEEE Trans. Reliab. 65 (2) (2016) 782–795.
[41] M. Shafiq, M. Atif, R. Viertl, Generalized likelihood ratio test and Cox's F-test based on fuzzy lifetime data, Int. J. Intell. Syst. 32 (1) (2017) 3–16.
[42] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[43] V. de Silva, J.B. Tenenbaum, Global versus local methods in nonlinear dimensionality reduction, in: NIPS, 2002, pp. 705–712.
[44] M. Kristan, A. Leonardis, D. Skocaj, Multivariate online kernel density estimation with Gaussian kernels, Pattern Recognit. 44 (10–11) (2011) 2630–2642.
[45] D.W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, John Wiley & Sons, 2015.
[46] P. Golik, P. Doetsch, H. Ney, Cross-entropy versus squared error training: a theoretical and experimental comparison, in: Interspeech, 2013, pp. 1756–1760.
[47] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: ICLR, 2014, pp. 1–15.

Fernando O. Gallego

Rafael Corchuelo