Take-Home Exam
A challenge set for SRL

Sharona Badloe
2712438

Abstract

This document contains the instructions for preparing a manuscript for the proceedings of ACL 2020. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used for both papers submitted for review and for final versions of accepted papers. Authors are asked to conform to all the directions reported in this document.

1 Introduction

In this report, a challenge set for Semantic Role Labeling (SRL) will be created with the use of CheckList. CheckList was created by a team of microsoft researchers, and is described as 'an evaluation methodology and accompanying tool" (Ribeiro et al., 2020). CheckList has been created to structure and improve the process of testing NLP models. In their introductory paper Ribeiro et al. (2020) argue that other testing approaches for NLP, such as evaluation metrics, adversarial changes and diagnostic datasets, are fragmented and incomplete. CheckList helps to create a comprehensive testing system that considers all relevant capabilities for your system in one place. The creators have been inspired by behavioral testing strategies applied in software engineering. Behavioral testing is also known as black-box testing, and is specifically used for testing complex models of which the exact internal structure is unknown, such as deep-learning models.

CheckList offers guidance by providing a list of linguistic capabilities that can be tested, which they state will be applicable to most tasks. These capabilities can the be tested with three different test types. The Minimum Functionality Test (MFT) is a collection of instances labeled with an expected result. The test simply checks if the system outputs the expected result for each instance or not. The Invariance test (INV) makes a minor change to each instance and checks if the predicted labels of the two paired instances stay the same. This test can be used to test situations where the data is modified but the system output should stay the same. The third option is a Directional Expectation test (DIR). Just like the INV test, this test also takes series of two instance pairs as input, except this time the labels should change. An expectation function should be used to specify the expected directional change.

2 Background to SRL

Semantic Role Labeling is the process of recovering the predicate-argument structure from a sentence (He et al., 2017). A predicate holds the central information of a situation that is expressed by a sentence, and the arguments portray the semantic roles relating to that predicate. In the task of SRL, one or more sentences are given as input. For each sentence, the predicate and its corresponding arguments are identified and labeled.

In the following section I will display some core capabilities that an SRL system should be able to handle. First I will go through some of CheckList's list of general capabilities, and (very briefly) discuss how they relate to SRL. Then I will mention some additional capabilities that I think are important for SRL.

2.1 Core capabilities of SRL

Complex sentence structures. Since SRL needs to take the entire sentence into account to correctly label the arguments, an SRL model should be able to deal with complex sentence structures. One of those complex structures are embedded clauses. An SRL system needs to know that an embedded clause often shares additional information on the noun phrase it is dependent on. This should not change the verb/argument structure.

Long-range dependency spans. This point relates to the preceding one: SRL needs to consider an entire sentence. So it could happen that an argument is very far removed from the predicate or other syntactic dependents. This results in a very long-range dependency span between the predicate and the argument that the SRL labeler needs to spot. It is important that an SRL model is able to deal with these long-range dependency spans.

Verb alternation. Verb alternation occurs when one verb can occur with various valency patterns. Valency, also known as complementation, refers to the type and number of arguments that a verb often occurs with or requires (Seyffarth, 2018). This is extremely important for SRL, because the task consists out of correctly identifying a verb as predicate, finding all its corresponding arguments and giving all arguments the correct semantic role. In order to spot the right arguments for the predicate, and assign them the correct role, an SRL system needs to be able to learn the syntactic conditions that determine which valency pattern accompanies a predicate in each sentence. In turn, it should consider this valency pattern when labeling the corresponding arguments.

Voice. The voice of the predicate will change the sentence and argument structure. SRL models need to be able know the differences between an active and passive verb and the relation to its arguments. In an active sentence, the sentence structure follows Subject + Verb + Object. In the case of SRL the subject will often be ARG0, agent, and the object will be ARG1, patient. A passive structure goes like the follows: Object + to be + Verb (+ by + Subject). Even though the structure is different, the subject and object should be identically labeled as ARG0 and ARG1 in both cases.

Figure of speech. Figure of speech is a very large part of our daily communication. Some forms of figure of speech are so ingrained in our language that we hardly notice it anymore. One of those figures of speech is metonymy. Metonymy occurs when we refer to a thing or concept by mentioning something that is strongly related to it. One variation of metonymy is synechdoche, when you refer to a whole by mentioning one of its parts. When metonymy occurs, the word that is used might have an entirely different tendency for semantic role labeling than the referent does. For example, using the term 'right hand' to refer to your assistant might fool an SRL system into thinking that this term cannot be labeled as ARG0, because a hand is not a sentient being and can therefore not be be an agent. However, because it is used as a figure of speech, the term 'right hand' directly refers to your assistant, who can definitely be an agent depending on the sentence. Since metonymy is such a large part of our everyday speech, a well-performing SRL system needs to be able to learn this.

Robustness. Robustness is an important part of every model in the field of Text Mining and AI. If your model is not robust, it means that it will easily be confused by minor changes which should not change the labeling task at hand. Since human data is often large and messy, a model should know which changes in language are important for its task, and which changes aren't. For SRL, adding typo's and negations should not change the argument labels. In order for an SRL system to provide consistent results, it should know to ignore or bypass these minor variations in data.

3 CheckList for SRL

In the following section, I will elaborate on the CheckList tests I have implemented. For each test I will first.

3.1 Long-range dependency spans

These checks aim to test how well the two SRL systems can correctly label an argument when the predicate and corresponding argument are far removed from each other. To test this, embedded clauses of different lengths are introduced into the sentence between the predicate and target argument. To learn more about how well our system does in relation to this capability, I have proposed two MFT tests. One focuses on the span between a predicate and one of its core roles, the other considers a predicate together with a non-core role. The difference in performance between these tests may give us additional information on what exactly the system struggles, or doesn't struggle with in relation to long-range dependency spans.

Test 1: Dependency span core role. This is an MFT test that explores the dependency span between an argument and one of its core roles. The example sentence that my challenge set for this test is based on is: 'The killer killed the victim with a knife.' In this sentence, 'killed' is the predicate, and 'with a knife' should get the label

ARG2, because it refers to the instrument used for the killing. ARG2 is a core role of the predicate, and is also our target argument for this test. An embedded clause will be inserted in between the predicate and the target argument, after which the MFT test checks if ARG2 is still labeled correctly.

To create a challenge set, I have augmented my example sentence with CheckLists Masked Language Model suggestion. I added a masked embedded clause into the example sentence to generate 100 examples. Each time the embedded clause is generated with different words in order to see if the semantics of the clause has an effect on correct labeling of the target argument. The clauses are also of different lengths in order to create a slow increase of span throughout my examples. This is done in order to see what the effect of the amount of words in a clause is on the correct labeling of ARG2 in our sentence.

Test 2: Dependency span non-core role. This is an MFT test that explores the dependency span between an argument and one of its non-core roles. This test has similar mechanisms to the first test in this capability. The only difference is in the sentence and the target argument. My baseline sentence in this test is: 'The killer killed the victim last week.' Here, 'last week' is not a core role of the predicate, but an ARG-TMP, since it is a clarification of when the killing happened. Embedded clauses with different words and of different lengths will be added to the baseline sentence to test if the longer span or certain semantics are confusing to the system.

3.2 Complex sentence structures

These checks aim to test how well the two SRL systems can correctly label an argument when a complex sentence structure is introduced. To test this, embedded clauses of different lengths are introduced into the sentence. This test set-up is similar to the ones that test long-range dependencies, but this time the embedded clause will not be put in between the predicate and the target argument. The embedded clause will be inserted in another place of the sentence, leaving the short distance between the predicate and the target argument intact. This is done to test specifically if embedded sentence structures confuse an SRL system, even when the dependency span is not an issue. Once again I have made the distinction of testing a predicate alongside a core role, and a predicate alongside a non-core role.

Test 1: Embedded clause core role. This is an MFT test that considers how well our SRL systems can identify a predicate and its core role in a complex sentence structure. The example sentence that my challenge set for this test is based on is: 'The killer killed the victim with a knife.' In this sentence, 'killed' is the predicate, and 'with a knife' should get the label ARG2, because it refers to the instrument used for the killing. ARG2 is a core role of the predicate, and is also our target argument for this test. An embedded clause will be inserted before the predicate, after which the MFT test checks if ARG2 is still labeled correctly.

Test 2: Embedded clause non-core role. This MFT test is similar to the one above, except it will consider a predicate together with a non-core role to see if there are any differences. The only difference is in the sentence and the target argument. My baseline sentence in this test is: 'The killer killed the victim last week.' Here, 'last week' is not a core role of the predicate, but an ARG-TMP, since it is a clarification of when the killing happened. Embedded clauses with different words and of different lengths will be added to the baseline sentence to see if the longer span is confusing to the SRL system.

3.3 Verb alternation

I will be exploring two types of verb alternation with my tests: dative alternation, and causative/inchoative alternation. As discussed above in section .., verb alternation refers to the valency pattern that verbs can have. My tests will focus on three types of valency: transitive, intransitive and ditransitive. Transitive verbs only require a direct object, intransitive verbs only require a subject and no objects, ditransitive verbs require a direct object and an indirect object. With the proposed tests below, I will try to gain more insight into how the two SRL models deal with some types of alternation.

Dative alternation. Dative alternation refers to the alternation between two types of syntactic constructions a ditransitive verb can occur in. The two types are indirect-object construction (The girl gave milk to the cat), and the double-object-

construction (The girl gave the cat milk). In both constructions, the SRL models need to correctly identify the subject, direct object and indirect object. I have decided to test this with two MFT tests, one for each structure. Both tests focus on the ditransitive verb 'to give' for direct comparison.

I have generated data for these tests with CheckLists masked language model, formats specified below. For evaluation, I wrote an expectation function that specifies that 'first_name1' should be labeled as ARG0 (agent/subject), 'mask' should be labeled as ARG1 (direct object/theme) and 'first_name2' should be labeled as ARG2 (indirect object/recipient). Only if all three arguments are labeled correctly is the test passed.

Causative-inchoative alternation. This type of alternation occurs with ambitransitive verbs, verbs that can be used both transitively and intransitively. If an ambitransitive verb occurs intransitively, we get the inchoative form that expresses the beginning of an action. When the verb is used transitively, we get the causative form that expresses a cause for the action. Even though the structure changes, the arguments should stay the same. To test this, I have created a directional test and generated causative/inchoative sentence pairs with CheckLists masked language model. The verb in focus is 'to break', and the masked word refers to the thing that is broken. I have written an expectation function that specifies that in both sentences in the pair, the masked word should be labeled as ARG1. If this is not the case, the test fails.

3.4 Voice

For my voice test I have generated some simple active/passive sentence pairs with different verbs. This is in order to test if the semantics of the verb has an effect on correct labeling. In short simple structures like this, the token in position of the mask will often get the label ARG1 (patient). I have created a DIR test that specifies that the mask in both sentences needs to be labeled as ARG1 to pass.

3.5 Robustness

Because the SRL systems performed so well on this simple active/passive sentence pair test, I have decided to perturb the data with typo's to see if the models still perform just as well on this task when minor changes are made to the data. This tests the robustness of the models to typo's.

4 Models

In this section, I will describe and compare the models that will be tested with my challenge set. Both models are taken from the Allen Institute for AI, a non-profit research institute (Gardner et al., 2017). They provide a library for natural language processing called AllenNLP. From this library I have taken two models to test: the standard srl model (He et al., 2017), and the srl BERT model (Shi and Lin, 2019).

4.1 SRL model

According to the AllenNLP documentation, this model is an implementation of the model described in the paper *What Works and What's Next* by He et al. (2017). This model consists out of 8 stacked bidirectional LSTM's and a softmax layer to make the predictions based on the probability of the LSTM outputs. It takes as input a sentence, after which the model identifies the predicate with an LSTM and softmax layer. This layer outputs a sentence-predicate pair which is then split up into tokens with a binary identifier specifying if the token is a predicate or not. These token-binary identifier pairs are then fed into the remainder of the system. There are a few constraints present in the model to guide the system into making the right predictions, by penalizing wrong formats. The BIO constraints are used to make sure that arguments and their BIO-labels are always displayed in the right order. SRL constraints make sure that rules specific to SRL are not broken, e.g. the same core role cannot appear more than once per predicate. Syntactic constraints reject arguments that are not constituents.

4.2 SRL BERT model

This model is an implementation of the paper by Shi and Lin (2019). This model assumes that the predicates will be given in the data, as that is the case in most of the CONLL shared tasks on SRL. The rest of the task is split up into predicate sense disambiguation, argument identification, and classification. A sentence is given to BERT's tokenizer, WordPiece, as input after which the predicate and remaining tokens are tagged. The resulting sentence-predicate pairs are represented in a way that captures the entire sentence context via

attention mechanisms, and then fed to the BERT encoder. This is followed by a BiLSTM to obtain hidden states, and finally an MLP (multilayer perceptron) classifier to make the predictions. The BERT model does not contain any human-generated constraints, but still performs excellently.

4.3 Comparison and expectations

Both models take a sentence as input which are then turned into sentence-predicate pairs. The srl model further splits up these sentence pairs into token-binary identifier pairs, whereas BERT takes the context of the sentence into account through attention mechanisms. The BERT model does not use any human-generated constraints whereas the standard srl does. Both models benefit from bidirectionality which is important for context.

5 Results

Complex sentence structures. Both systems did not do very well with either dependency spans or embedded clauses. A core role would often be mislabeled as a non core role when an embedded clause was introduced. For example: The killer who pleaded not guilty to murder, killed the victim with a knife. 'with a knife' here should be labeled as ARG2 because it is a core role of the predicate 'to kill'. The sentence without an embedded clause is labeled correctly. Once an embedded clause is introduced, 'with a knife' is labeled as ARGM-MNR. Other mistakes include labeling 'the victim with a knife' as patient. Both systems had 100%, or almost 100% of examples incorrect. The SRL had the best result with 96% for non-core role dependency spans.

Verb alternation. The inchoative/causative test went well on both systems. The SRL model only had 10% of examples incorrect, where most of the wrong examples were due the template generating a sentient being in the place of the target argument. In the inchoative structure, the system then confused this for an agent. For example: Mary broke the twins. The twins broke. Here the twins was wrongly labeled as ARG0, when they are actually supposed to have the same label as 'the twins' in the causative structure: ARG1. The srl BERT model did not have this problem and labeled all causative/inchoative pairs correctly.

The dative alternations also performed quite well. The SRL BERT model had 14% of DOC and 20% IOC alternations wrong. The SRL model had 20% of DOC and 21% of IOC alternations incorrect. The fails in both systems seem to have the similar sources. For example: [ARG0: Sally] [V: gave] [ARG2: Robert CPR]. Here the BERT system confused CPR with some sort of last name, rather than an object that was given.

Voice. The systems did not have trouble with my active/passive pairs. the SRL BERT model had all of them correct, despite typo's. The SRL model had 2% of examples wrong, both of them involving the verb 'heard'. All other verbs in the generated examples had no issues.

Robustness. Both systems passed the robustness test with 0% errors.

6 Discussion

It is important to realize the limitations of the CheckList tests when evaluating the results. The tests in the capabilities Complex Sentence Structure and Dependency Span, only check the predicate with either an ARG2 or prepositional argument. It does not consider ARG0 org ARG1, which could give us very different results. There are also many other complex sentence structures besides a single embedded clause that could be considered in further research. For these two capabilities, my expectation was that the systems would have more of a problem with the long-range dependence, than with an embedded clause that does not influence the dependency range of the target argument. However, the opposite seems to be the case. Both systems are more confused by an embedded clause before the predicate, than they are by an embedded clause between the predicate and target argument. This is perhaps because the range between the agent and the predicate is now bigger which influences how the system interprets other arguments, as well as the target argument (ARG2/ARG-TMP).

The directional tests, causative/inchoative and active/passive, only check if the target argument is the same, not what the target argument is supposed to be. This is because different generated words might change the semantics, and therefore the argument structure of the sentence. It is hard to pin down in a long generated list of examples what the target argument is supposed to be. However, when taking this into account, this test can still give us some information about our systems. In the active/passive test for example, the test tells us

if the masked word in both sentence structures gets the same label. Therefore we find out if the system is able to recognize that the masked word in these two positions are the same argument. When this response is correct, more tests can be done to figure out if the system also gives both words the right label.

7 Future Work

This section contains some suggestions for using the contents of this report for a domain adaptation task involving SRL. The example task aims to test how well an SRL model trained on the universal PropBank frames will perform on domain-specific data from the medical domain. Two medical datasets will be provided: a small corpus of labeled data and a larger corpus of unlabeled data. I will approach this example task by creating a challenge set using CheckList, augmented from the domain-specific corpora.

First, I would perform an extensive data analysis of the unlabeled data by means of running a dependency parser on the full corpus. From this parser I would extract the most frequent constituents, split by constituent type. I would end up with a list of domain-specific noun phrases, verb phrases and prepositional phrases, along with the predicates which can easily be extracted by getting the head of the main clause in a sentence. These domain-specific constituents can then be used as masked suggestions for CheckLists Masked Language Model.

Now that we have lists of medical constituents, they can be used for data generation using templates. The template formats will be selected from the labeled corpus. I would choose sentences based on the domain-specific traits I would like to test. It is important to make sure that any masked modifications that are generated do not change the label of the target argument that is tested with each sentence format. Both the semantics of the predicate and the syntactic structure can change the label of the corresponding argument. I suggest simple changes per format to make sure our valuable gold labels are not lost.

The types of tests that this template-generated data can be used for depends on the domain, and the capabilities and limitations of your SRL model. First a few standard NLP capabilities can be studied. Then, a few tests that are more specific to SRL could be implemented, not unlike the tests shown in this report. Finally, a few domain-specific tests would need to be considered. For example, data from the medical domain has a tendency to contain a lot of obscure medicine names that a standard SRL system would not have seen before. To test how an SRL responds to this, a labeled sentence can be selected from the data. Such as, 'The doctor determined that dexamfetamine was performing well for the patient.' The medicine in this sentence could be replaced with a [mask], and words could be suggested from our list of frequent medical noun phrases. We could consider only the one-token noun phrases in this case. When enough examples are generated, some other medicines would be among those frequent noun phrases and would likely be inserted in place of the original medicine. A template like this could be used to test if the SRL system can correctly identify medicine names, or if it would mistake the medicine for a person name and label it as agent. I have structured the example sentence in a way that both a sentient or non-sentient being could take the position of the [mask]. It thus depends on how well the SRL is able to spot the semantics of each medicine if the argument is still labeled correctly.

Not many labeled sentences would have to be selected from the corpus. Hopefully, this report has shown that even with as little as eight different sentence templates, quite a few reasonably informative tests can be implemented. With more time and research, a small but valuable set of sentences can be extracted from the labeled data to generate examples for a number of well-thought out tests. A 100 well-chosen labeled sentences could give you a 100 (or more) tests, each with as many generated examples as you deem necessary. Depending on how well the tests are structured, this should be more than sufficient to test all important general and domain-specific SRL traits. Depending on how large your labeled corpus is, after extracting the sentences you want to use for example generation for a challenge set, the rest of the sentences can be used as extra evaluation set. Depending on how your SRL works, this remainder of labeled data can be used as test set and evaluated with the standard precision, recall and f1 metrics. If the SRL system only takes one sentence at a time, the remainder of the labeled sentences can added to the challenge set for use in an MFT test.

Alternatively, instead of running a dependency

parser to get the most frequent constituents for CheckLists masked language suggestion, an separate language model, such as BERT, could be fine-tuned on the unlabeled corpus. This fine-tuned model could then make predictions for masked positions in the template sentences. The created examples can then be used in a number of tests (CheckList or otherwise) in the same way. I suspect that this method would give more accurate suggestions, as CheckLists masked language model is not domain-specific, and will therefore not put the correct weights on the most frequent constituents that are manually put in the suggestions list in the method described above. However, the difference might not be large enough for this to be necessary.

8 Conclusion

In this report, a challenge set was created with CheckList in order to test and compare two different Semantic Role Labelers on their performance. Five different SRL capabilities were tested with the use of nine different CheckList tests. The tested capabilities can be found in table 1 below.

Overall, the SRL BERT model outperformed the SRL model, but not by much. It seems like the BERT model is not as easily confused by context words, and does a better job at taking semantics and the effect on the argument structure into account.

References

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Esther Seyffarth. 2018. Verb alternations and their impact on frame induction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 17–24, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Peng Shi and Jimmy J. Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255.

A Appendix

Table 1: Test overview

Capability	Type	Test name	Target	Example	Template
Long-range dependency spans	MFT	Span core role	ARG2	The killer killed the victim, who declared herself as mentally unstable, [with a knife] ARG2	'The killer killed the victim, {mask} {mask} {mask} {mask} {mask} {mask}, with a knife.'
	MFT	Span non-core role	ARG-TMP	The killer killed the victim, who always went out for dinner, [last week] ARG-TMP	'The killer killed the victim, {mask} {mask} {mask} {mask} {mask} {mask}, last week.'
Complex sentence structures	MFT	Complex core role	ARG2	The killer, who was late for school yesterday, killed the victim [with a knife] ARG2	'The killer, {mask} {mask} {mask} {mask} {mask} {mask}, killed the victim with a knife.'
	MFT	Complex non-core role	ARG-TMP	The killer, who never listened to his parents, killed the victim [last week] ARG-TMP	'The killer, {mask} {mask} {mask} {mask} {mask} {mask}, killed the victim last week.'
Verb alternation	MFT	Dative alternation IOC	ARG0 ARG1 ARG2	[ARG0: Maria] [V: gave] [ARG1: books] [ARG2: to Jim] first_name1 = ARG0 first_name2 = ARG2 mask = ARG1	{first_name1} gave {mask} to {first_name2}
	MFT	Dative alternation DOC	ARG0 ARG1 ARG2	[ARG0: Frances] [V: gave] [ARG2: Jill] [ARG1: copies] first_name1 = ARG0 first_name2 = ARG2 mask = ARG1	{first_name1} gave {mask} to {first_name2}
	DIR	Causative / inchoative	ARG1	He broke [the vase] / [The vase] broke The vase = ARG1	Mary broke the {mask1} / The {mask1} broke
Voice	DIR	Active/passive pairs	ARG1	[ARG1: The review] was [V: written] / She [V: wrote] [ARG1: The review] The review = ARG1	The {mask1} was written / She wrote the {mask1}
Robustness	DIR	Active/passive pairs	ARG1	[ARG1: The review] was [V: wirtten] / She [V: wrote] [ARG1: Teh review] The review = ARG1	Data for voice test perturbed with typo's