APPENDIX B: Extracting Significant Information (Subject-Verb-Object – SVO) From Dependency Parsing

In this step, the sentence structure and its *typed dependencies* (TDs) are analyzed to identify *noun* and *verb* phrases in *subject*, *object* and *action-verb* roles. For doing so, patterns that take into account a subset of *typed dependencies* (*td_name*(*gov*, *dep*)) in the sentence, POS tags of the *head* (*gov*) and *dependent* (dep) words of a given *typed dependency* are used. To extract these patterns, a study was performed with existing input projects. The proposed approach uses Stanford CoreNLP Parser version 3.9.2 to generate TDs from the sentences. The Stanford CoreNLP parser uses the *Universal Dependencies* (https://universaldependencies.org/) representation.

Fig. 1 illustrates a few rules for extracting subjects, action-verbs, direct-objects and indirect-objects from typed dependencies.

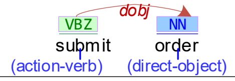
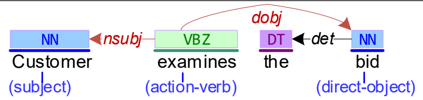
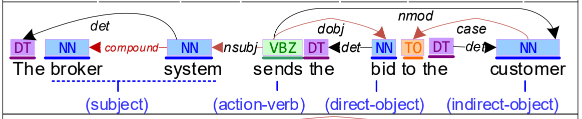
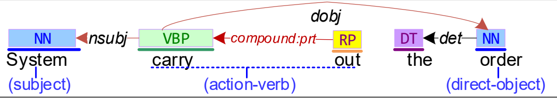
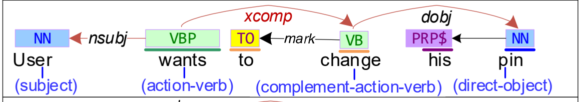
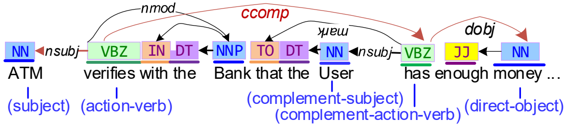
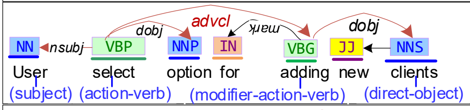
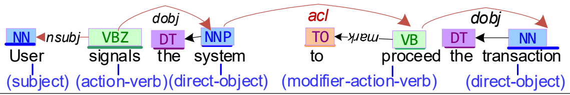
Sentence	Typed Dependecies				Extract Subject, Object and Action-Verb from Typed Dependencies
	td-name	gov	dep	Rule description	
Submit order	*dobj*	submit-1	order-2	"order" is direct object of "submit"	
Customer examines the bid	*nsubj* *dobj* *det*	examines-2 examines-2 bid-4	customer-1 bid-4 the-3	"customer" is subject of "examines" "bid" is direct object of "examines" "the" is determiner of "bid"	
The broker system sends the bid to the customer	*nsubj* *dobj* *compound* *nmod* ...	sends-4 sends-4 system-3 sends-4 ...	system-3 bid-6 broker-2 customer-9 ...	"system" is subject of "sends" "bid" is direct object of "sends" "broker" is compund of "system" "customer" is indirect object of "sends" ...	
System carry out the order	*nsubj* *dobj* *compound:prt* ...	carry-2 carry-2 carry-2 ...	system-1 order-5 out-3 ...	"system" is subject of "carry" "order" is direct object of "carry" "out" is particle of "carry" ...	
User wants to change his PIN	*nsubj* *dobj* *mark* *xcomp* ...	wants-2 change-4 change-4 wants-2 ...	user-1 pin-6 to-3 change-4 ...	"user" is subject of "wants" "pin" is direct object of "change" "to" introduces the clause "change" "change" is complement "TO" of "wants" ...	
ATM verifies with the Bank that the User has enough money in account	*nsubj* ... *ccomp* *nsubj* *dobj* ...	verifies-2 ... verifies-2 has-9 has-9 ...	ATM-1 ... has-9 user-8 money-11 ...	"ATM" is subject of "verifies" ... "has" is complement-verb of "verifies" "user" is complement-subject of "has" "money" is direct object of "has" ...	
User select option for adding new clients	*nsubj* ... *advcl* ...	select-2 ... select-2 ...	User-1 ... adding-5 ...	"User" is subject of "select" ... "adding" is modifier-verb of "select" ...	
User signals the system to proceed the transaction	*nsubj* ... *acl* ...	signals-2 ... system-4 ...	User-1 ... proceed-6 ...	"User" is subject of "select" ... "adding" is modifier-verb of "select" ...	

Fig. 1. Information extraction using NLP dependency parsing.

Tables 1, 2, 3 and 4 detail the rules to extract the *subjects*, *objects* and *action-verbs* in a given sentence. These rules are presented in GIVEN-WHEN-THEN format and they are ordered. The approach sequentially searches for all TDs in the GIVEN (Antecedent) part of each rule against the TDs of the given sentence. In case of *antecedent* is satisfied, the WHEN part of the rule is activated; the approach sequentially checks the POS tags of *gov* and *dep* words (A, B, C, D) of the given TDs. In case of a match, the THEN (Consequent) part of the rule is used to determine the structure of that sentence and extract the *subjects*, *objects* and *action-verbs*.

In Tables 1, 2, 3 and 4, *subjects* = { $token_i$, $token_{i+1}$, …, $token_{n-1}$, $token_n$} is the set of tokens with subject role in a given sentence, *direct-objects* = {$token_i$, $token_{i+1}$, …, $token_{m-1}$, $token_m$} is the set of tokens with direct-object role, *indirect-objects* = {$token_i$, $token_{i+1}$, …, $token_{p-1}$, $token_p$} is the set of tokens with indirect-object role, *action-verbs* = {$token_i$, $token_{i+1}$, …, $token_{q-1}$, $token_q$} is the set of tokens with main-action-verb role, *complement-action-verbs* = {$token_i$, $token_{i+1}$, …, $token_{r-1}$, $token_r$} is the set of tokens with complement-action-verb role, *modifier-action-verbs* = {$token_i$, $token_{i+1}$, …, $token_{s-1}$, $token_s$} is the set of tokens with modifier-action-verb role, *modifier-subjects* = {$token_i$, $token_{i+1}$, …, $token_{t-1}$, $token_t$} is the set of tokens with modifier-subject role, *complement-subjects* = {$token_i$, $token_{i+1}$, …, $token_{r-1}$, $token_r$} is the set of tokens with complement-subject role, $token_i$ = {*index*, *word*, *POS*, *lemma*} is a token and its properties (word and lemma are either a single word or a multiword), () is used for grouping, | stands for "OR", ? matches the preceding character 0 or 1 time, and "." matches any single character except the newline character.

TABLE 1
RULES FOR EXTRACTING SUBJECT, OBJECT AND ACTION-VERBS FROM TYPED DEPENDENCIES – (SVO FROM SINGLE-WORDS).

Rule #	*Description*	GIVEN (Contains TDs)	WHEN (Contains POSs:)				THEN (Extract Role)	Example
			A	*B*	*C*	*D*		

SVOR1	Extract *subject* and *action-verb* from *nsubj* relationship	nsubj(A, B)	VB.?	(NN.? | PRP.?)			*A* is action-verb;
							B is subject;
								"**Customer examines** the bid" → [**nsubj**(**examines**-2, **customer**-1), root(ROOT-0, examines-2), det(bid-4, the-3), dobj(examines-2, bid-4)]

SVOR2	Extract *subject*, *direct-object* and *action-verb* from *nsubjpass* and *nmod* (or dobj) relationships	nsubjpass(A, B), nmod(A, C)	VB.?	(NN.? | PRP.?)	(NN.? | PRP.? | WP.?)		*A* is action-verb;
							B is direct-object;
							C is subject;
								"File was **updated** by the **user"** → [**nsubjpass**(**updated**-3, **File**-1), auxpass(updated-3, was-2), root(ROOT-0, updated-3), **case**(user-6, by-4), det(user-6, the-5), **nmod**(updated-3, **user**-6)]

SVOR3	Extract *direct-object* and *action-verb* from *dobj* relationship	dobj(A, B)	(JJ | VB.?)	(NN.? | PRP.?)			*A.POS* ≠ JJ → *A* is action-verb;
							B is direct-object;
								"User creates **filter** for searching" →[nsubj(creates-2, user-1), root(ROOT-0, creates-2), **dobj**(**creates**-2, **filter**-3), mark(searching-5, for-4), advcl(creates-2, searching-5)]

SVOR4	Extract *indirect-object* from *iobj* relationship	iobj(A, B)	(JJ | VB.?)	(NN.? | PRP.?)			*B* is indirect-object;	**"**System **sends** the **server** a registration request**"** → [**nsubj**(sends-2, system-1), root(ROOT-0, sends-2), det(server-4, the-3), **iobj**(sends-2, **server**-4), det(request-7, a-5), compound(request-7, registration-6), **dobj**(sends-2, request-7)]

SVOR5	Extract *subject*, *action-verb* and *indirect-object* from *nsubj* and *nmod* relationships	nsubj(A, B), nmod(A, C)	VB.?	(NN.? | PRP.?)	(NN.? | PRP.?)		*A* is action-verb;
							B is subject;
							C is indirect-object;
								"**User clicks** on the **screen**" → [**nsubj**(**clicks**-2, user-1), root(ROOT-0, clicks-2), **case**(screen-5, on-3), det(screen-5, the-4), **nmod**(clicks-2, **screen**-5)]

		nmod(A, B)	VB.?	(NN.? | PRP.?)			*nsubjpass*(A, ?) ∉ TDs →
							A is action-verb;
							B is indirect-object;
								"**Log** in to the **system**" → [**root**(**ROOT**-0, **log**-1), case(system-5, in-2), case(system-5, to-3), det(system-5, the-4), **nmod**(**log**-1, **system**-5)]

SVOR6	Extract *subject*, *action-verb* and *indirect-object* from *dobj* and *nmod* relationships	dobj(A, B), nmod(B, C)	VB.?	(NN.? | PRP.?)	(NN.? | PRP.?)		*A* is action-verb;
							B is direct-object;
							C is indirect-object;
								"**User clicks** the **mouse** on the **screen**" → [nsubj(clicks-2, user-1), root(ROOT-0, clicks-2), det(mouse-4, the-3), **dobj**(**clicks**-2, **mouse**-4), **case**(screen-7, on-5), det(screen-7, the-6), **nmod**(**mouse**-4, **screen**-7)]

SVOR7	Extract *action-verb* from *ROOT* relationship	root(ROOT, A)	VB.?				*A* is action-verb;	**"selects** envelope**"** → [**root**(ROOT-0, selects-1), nsubj(selects-1, envelope-2)]

SVOR8	Update *subject*, *direct-object* or *indirect-object* from *case* and *nmod* relationships	case(A, of), nmod(B, A)	NN.?	NN.?	NN.?		*B* is subject → *Remove* *B*; *A* is subject;
							B is direct-object → *Remove* *B*; *A* is direct-object;
							B is indirect-object → *Remove* *B*; *A* is indirect-object;
								"system **displays** set **of** possible **criteria**" → [nsubj(displays-2, system-1), root(ROOT-0, displays-2), dobj(displays-2, **set**-3), **case**(**criteria**-6, of-4), amod(criteria-6, possible-5), **nmod**(**set**-3, **criteria**-6)]

TABLE 2
RULES FOR EXTRACTING SUBJECT, OBJECT AND ACTION-VERBS FROM TYPED DEPENDENCIES – (SVO FROM MULTI-WORDS).

Rule #	***Description***	**GIVEN (Contains TDs)**	**WHEN (Contains POSs:)**				**THEN (Extract Role)**	**Example**
			A	*B*	*C*	*D*		
SVOR9	Update multiword *subject*, *direct-object or indirect-object* from *compound* relationship	compund(A, B)	NN.?	NN.?			***A*** is subject → *Update* ***A*** *=* ***B+A****;* ***A*** is direct-object → *Update* ***A*** *=* ***B+A****;* ***A*** is indirect-object → *Update* ***A*** *=* ***B+A****;*	"The **broker system** broadcasts the order" →[det(system-3, the-1), **compound**(**system**-3, **broker**-2), nsubj(broadcasts-4, system-3), root(ROOT-0, broadcasts-4), det(order-6, the-5), dobj(broadcasts-4, order-6)]
SVOR10	Update multiword *subject*, *direct-object or indirect-object* from *nmod:poss* relationship	nmod:poss(A, B)	NN.?	NN.?			***A*** is subject → *Update* ***A*** *=* ***B*+ POSSESIVE +** ***A****;* ***A*** is direct-object → *Update* ***A*** *=* ***B*+ POSSESIVE +** ***A****;* ***A*** is indirect-object → *Update* ***A*** *=* ***B*+ POSSESIVE +** ***A****;*	"Broker system broadcasts **customer's information**" → [compound(system-2, broker-1), nsubj(broadcasts-3, system-2), root(ROOT-0, broadcasts-3), **nmod:poss**(**information**-6, **customer**-4), **case**(**customer**-4, '**s**-5), dobj(broadcasts-3, information-6)]
SVOR11	Update multiword *subject*, *direct-object or indirect-object* from *nummod* relationship	nummod(A, B)	NN.?	CD.?			**IF** ***A****.index <* ***B****.index* → ***C*** *=* ***A+B****;* **ELSE** ***C*** *=* ***B+A****;* ***A*** is subject → *Update* ***A*** *=* ***C****;* ***A*** is direct-object → *Update* ***A*** *=* ***C****;* ***A*** is indirect-object → *Update* ***A*** *=* ***C****;*	"System returns to **step 1.1** " →[root(ROOT-0, system-1), dep(system-1, return-2), case(step-5, to-3), det(step-5, the-4), nmod(return-2, step-5), **nummod**(**step**-5, **1.1**-6)]
SVOR12	Update multiword *action-verb* from *compound:prt* relationship	compound:prt(A, B)	VB.?	RP.?			***A*** is action-verb → *Update* ***A*** = ***A+B****;*	"The broker system **carry out** the order" →[det(system-3, the-1), compound(system-3, broker-2), nsubj(carry-4, system-3), root(ROOT-0, carry-4), **compound:prt**(**carry**-4, **out**-5), det(order-7, the-6), dobj(carry-4, order-7)]

TABLE 3
RULES FOR EXTRACTING SUBJECT, OBJECT AND ACTION-VERBS FROM TYPED DEPENDENCIES – (SVO FROM SUBORDINATE OR COORDINATE).

Rule #	*Description*	**GIVEN (Contains TDs)**	**WHEN (Contains POSs:)**				**THEN (Extract Role)**	**Example**
			A	*B*	*C*	*D*		
SVOR13	Update *action-verb* from *xcomp* relationship	root(ROOT, A), xcomp(A, B)	VB.?	VB.?			***B*** is action-verb → *Remove **B**;* ***B*** is complement-acion-verb;	"User **wants** to **change** his pin" → [nsubj(wants-2, user-1), root(ROOT-0, wants-2), mark(change-4, to-3), **xcomp**(**wants**-2, **change**-4), nmod:poss(pin-6, his-5), dobj(change-4, pin-6)]

SVOR14	Update *action-verb* from *ccomp* relationship	root(ROOT, A), nsubj(B, C), ccomp(D, B)	VB.?	VB.?	(<NN.?> | <PRP.?> | WP.?)	<VB.?>	***B*** is action-verb → *Remove **B**;*
							B is complement-acion-verb;
							C is subject → *Remove **C**;*
							C is complement-subject;
								"ATM **verifies** with the Bank that the **User has** enough money in account"
								→ [nsubj(verifies-2, atm-1), **root**(ROOT-0, **verifies**-2), case(bank-5, with-3), det(bank-5, the-4), nmod(verifies-2, bank-5), mark(has-9, that-6), det(user-8, the-7), **nsubj**(**has**-9, **user**-8), **ccomp**(**verifies**-2, **has**-9), amod(money-11, enough-10), dobj(has-9, money-11), case(account-13, in-12), nmod(money-11, account-13)]

		nsubjpass(A, B), ccomp(C, A), nmod(A, D)	VB.?	(<NN.?> | <PRP.?> | WP.?)	<VB.?>	(<NN.?> | <PRP.?> | WP.?)	***A*** is action-verb → *Remove **A**;*
							A is complement-acion-verb;
							D is subject → *Remove **D**;*
							D is complement-subject;
								"System displays an information that it cannot be used without prior registration"
								→ [nsubj(displays-2, system-1), root(ROOT-0, displays-2), det(information-4, an-3), dobj(displays-2, information-4), mark(used-10, that-5), **nsubjpass**(**used**-10, **it**-6), aux(used-10, can-7), neg(used-10, not-8), auxpass(used-10, be-9), **ccomp**(**displays**-2, **used**-10), case(registration-13, without-11), amod(registration-13, prior-12), **nmod**(**used**-10, **registration**-13)]

SVOR15	Update *action-verb* from *advcl* relationship	advcl(A, B)	VB.?	VB.?			***B*** is action-verb → *Remove **B**;*
							B is modifier-acion-verb;
							A is acion-verb;
								"user select option for adding new clients"
								→ [**nsubj**(select-2, user-1), root(ROOT-0, select-2), dobj(select-2, option-3), mark(adding-5, for-4), **advcl**(**select**-2, **adding**-5), amod(clients-7, new-6), dobj(adding-5, clients-7)]

SVOR16	Update *subject*, *direct-object* and *action-verb* from *root*, *nsubj*, and *advcl* relationships	root(ROOT, A), nsubj(B, C), advcl(A, B)	VB.?	(NN.? | PRP.?)	(NN.? | PRP.? | WP.?)		***A*** is action-verb → *Remove A;*
							A is modifier-acion-verb;
							B is modifier-subject;
							C is direct-object;
								"Use case **ends** when **user logs** out or **selects** different option"
								→ [compound(case-2, use-1), nsubj(ends-3, case-2), **root**(ROOT-0, **ends**-3), advmod(logs-6, when-4), **nsubj**(**logs**-6, **user**-5), **advcl**(**ends**-3, **logs**-6), compound:prt(logs-6, out-7), cc(logs-6, or-8), conj(logs-6, selects-9), amod(option-11, different-10), dobj(selects-9, option-11)]

								"System **asks** the user if he/**she wants** to register" → [nsubj(asks-2, system-1), **root**(ROOT-0, **asks**-2), det(user-4, the-3), dobj(asks-2, user-4), mark(wants-7, if-5), **nsubj**(**wants**-7, **she**-6), **advcl**(**asks**-2, **wants**-7), mark(register-9, to-8), xcomp(wants-7, register-9)]

SVOR17	Update *action-verb* from *acl* relationship	acl(A, B)	NN.?	VB.?			***B*** is action-verb → *Remove B;*
							B is modifier-acion-verb;
								"user **signals** the system to **proceed** the transaction"
								→ [nsubj(signals-2, user-1), root(ROOT-0, signals-2), det(system-4, the-3), **dobj**(signals-2, system-4), mark(proceed-6, to-5), **acl**(**system**-4, **proceed**-6), det(transaction-8, the-7), dobj(proceed-6, transaction-8)]

SVOR18	Update *subject*, *direct-object* and *action-verb* from *nsubjpass*, *dobj* and *acl*	dobj(A, B), nsubjpass(A, C), *acl*(D, A),	VB.?	(NN.? | PRP.?)	(NN.? | PRP.? | WP.?)	NN.?	***A*** is action-verb → *Remove A;*
							A is modifier-acion-verb;
								"User select a client for **whom** new **contract** will be **added**"
								→ [nsubj(select-2, user-1), root(ROOT-0, select-2), det(client-4, a-3), dobj(select-2, client-4), mark(added-11, for-5),

	relationships						***B*** is modifier-subject;
							C is direct-object;
								dobj(**added**-11, **whom**-6), amod(contract-8, new-7), **nsubjpass**(**added**-11, **contract**-8), aux(added-11, will-9), auxpass(added-11, be-10), **acl**(**client**-4, **added**-11)]

SVOR19	Update *action-verb* from *acl:relcl* relationship	nsubj(A, B), acl:relcl(C, A)	VB.?	(NN.? | PRP)	(NN.? | PRP | WP.?)		***A*** is action-verb → *Remove **A**;*
							A is modifier-acion-verb;
							B is subject → *Remove **B**;*
							B is modifier-subject;
								“Administrator chooses a group containing the channel **he wants** to delete”
								→ [nsubj(chooses-2, administrator-1), root(ROOT-0, chooses-2), det(group-4, a-3), dobj(chooses-2, group-4), acl(group-4, containing-5), det(channel-7, the-6), dobj(containing-5, channel-7), **nsubj**(**wants**-9, **he**-8), **acl:relcl**(**channel**-7, **wants**-9), mark(delete-11, to-10), xcomp(wants-9, delete-11)]

TABLE 4
Rules For Extracting Subject, Object and Action-Verbs From Typed Dependencies – (SVO from Conjuntion).

Rule #	***Description***	**GIVEN (Contains TDs)**	**WHEN (Contains POSs:)**				**THEN (Extract Role)**	**Example**
			A	*B*	*C*	*D*		

SVOR20	Extract *subject*, *direct-object or indirect-object* from *conj* relationship	conj(A, B),	(NN.? | PRP.?)	(NN.? | PRP.?)			***A*** is subject → ***B*** is subject;
							A is direct-object → ***B*** is direct-object;
							A is indirect-object → ***B*** is indirect-object;
								“User informs their **login** and **password**”
								→[nsubj(informs-2, user-1), root(ROOT-0, informs-2), nmod:poss(login-4, their-3), dobj(informs-2, login-4), cc(login-4, and-5), **conj**(**login**-4, **password**-6)]

SVOR21	Extract *action-verb*, *complement-action-verb or modifier-action-verb* from *conj* relationships	conj(A, B),	VB.?	VB.?			***A*** is action-verb → ***B*** is action-verb;	“User **register** or **delete** transactions” → [nsubj(register-2, user-1), root(ROOT-0, register-2), cc(register-2, or-3), **conj**(**register**-2, **delete**-4), dobj(delete-4, transactions-5)]