Software-defined Network Assimilation: Bridging the Last Mile Towards Centralized Network Configuration Management with NAssim

Huangxun Chen¹, Yukai Miao², Li Chen³, Haifeng Sun⁴, Hong Xu⁵, Libin Liu⁶, Gong Zhang⁷, Wei Wang⁷

¹Huawei Theory Lab, ²University of New South Wales, ³Zhongguancun Laboratory
⁴Beijing University of Posts and Telecommunications, ⁵The Chinese University of Hong Kong
⁶Shandong Computer Science Center (National Supercomputer Center in Jinan)
⁷Hong Kong University of Science and Technology (Guangzhou & Hong Kong)
Applications
The ideal network for NetOps consists of homogeneous devices

The ideal network (Homogeneous Device Model)
The ideal network for NetOps consists of homogeneous devices
Applications

[Diagram of applications and cloud services]
The real network consists of multi-vendor heterogeneous devices

Applications

The real network (Heterogeneous Device Model)
The real network consists of multi-vendor heterogeneous devices

Applications

Competition between multiple vendors

The real network (Heterogeneous Device Model)
The real network consists of multi-vendor heterogeneous devices

Applications

- Competition between multiple vendors
- Coexistence of legacy & latest devices

The real network (Heterogeneous Device Model)
The real network consists of multi-vendor heterogeneous devices

Applications

Competition between multiple vendors
Coexistence of legacy & latest devices

The real network (Heterogeneous Device Model)
Managing a multi-vendor network is a pain...
Managing a multi-vendor network is a pain...

SDN Controller
Managing a multi-vendor network is a pain...

Unified Device Model (UDM)

SDN Controller

Expert

create

BGP

AS Number

BFD Session

-Link Bundle

BFD Rx

BFD Multiplier
Managing a multi-vendor network is a pain...

Unified Device Model (UDM)

- Expert
- BGP
- AS Number
- BFD Rx
- BFD Multiplier

create

match

manual

SDN Controller
Managing a multi-vendor network is a pain...

**Unified Device Model (UDM)**

- **create**
- BGP
- AS Number
- BFD Rx
- BFD Multiplier

**SDN Controller**

- **match**
- manual
- read manuals
- create mappings

**CLI command**

```
system-view
bgp <as-number>
```

```
system-view
bfd <session-name> bind peer-ip <peer-ip>
detect-multiplier <multiplier>
```
Managing a multi-vendor network is a pain...

Unified Device Model (UDM)

Expert

create

BGP

AS Number

BFD Rx

BFD Multiplier

BFD Session - Link Bundle

SDN Controller

match

manual

read manuals

create mappings

CLI command

system-view

bgp <as-number>

CLI command

system-view

bfd <session-name> bind peer-ip <peer-ip>
detect-multiplier <multiplier>
Managing a multi-vendor network is a pain...

Unified Device Model (UDM)

Expert

create

BGP

AS Number

BFD Session

-Link Bundle

BFD Rx

BFD Multiplier

SDN Controller

match

read manuals
create mappings

Expert

system-view

bgp <as-number>

CLI command

system-view

lag <lag-id>
bfd
family { ipv4 | ipv6 }
multipilier [ <multiplier> ]

CLI command

read manuals
create mappings

Expert

CLI command

service
vprn <vprn-id> customer <cid> create
automous-system <as-number>

CLI command
Managing a multi-vendor network is a pain...

**Unified Device Model (UDM)**
- create
- BGP
- AS Number
- BFD Rx
- BFD Multiplier

**SDN Controller**
- read manuals
- create mappings

**CLI command**
- system-view
- bgp <as-number>

**CLI command**
- system-view
- bfd <session-name> bind peer-ip <peer-ip> detect-multiplier <multiplier>

**Expert**
- read manuals
- create mappings

**CLI command**
- service
- vprn <vprn-id> customer <cid> create
- autonomous-system <as-number>

**CLI command**
- system-view
- lag <lag-id>
- bfd
  - family { ipv4 | ipv6 }
  - multiplier [ <multiplier> ]
Current SNA approaches require significant human efforts
Current SNA approaches require significant human efforts

Software-defined Network Assimilation (SNA) - The process of introducing heterogeneous network devices (e.g., legacy devices & devices from a new vendor) into a centrally controlled, existing SDN network.
Current SNA approaches require significant human efforts

**Unified Device Model (UDM)**

**Vendor-specific Device Models (VDM)**

*Software-defined Network Assimilation (SNA)*

The process of introducing heterogeneous network devices (e.g., legacy devices & devices from a new vendor) into a centrally controlled, existing SDN network.
Current SNA approaches require significant human efforts

Software-defined Network Assimilation (SNA)

The process of introducing heterogeneous network devices (e.g., legacy devices & devices from a new vendor) into a centrally controlled, existing SDN network.

Unified Device Model (UDM)

Vendor-specific Device Models (VDM)

Tedious Error-prone
Current SNA approaches require significant human efforts

Software-defined Network Assimilation (SNA)

The process of introducing heterogeneous network devices (e.g., legacy devices & devices from a new vendor) into a centrally controlled, existing SDN network.

Unified Device Model (UDM)

Vendor-specific Device Models (VDM)

Tedious Error-prone

Automated Efficient
Basic Insight: Imitating the Practices of NetOps

**Vendor-specific Device Models (VDM)**

**Unified Device Model (UDM)**
Basic Insight: Imitating the Practices of NetOps

Vendor-specific Device Models (VDM)

Preliminary Device Model

Refined and Validated Device Model

Unified Device Model (UDM)
Basic Insight: Imitating the Practices of NetOps

Vendor-specific Device Models (VDM)

Preliminary Device Model

Refined and Validated Device Model

Unified Device Model (UDM)

CLI

bgp <as-number>
peer <ipv4-address> group <group-name>

semantics

JSON
Challenges for SDN Network Assimilation (SNA)

Vendor-specific Device Models (VDM)

Preliminary Device Model

Refined and Validated Device Model

Unified Device Model (UDM)
Challenges for SDN Network Assimilation (SNA)

Vendor-specific Device Models (VDM)

Preliminary Device Model

Refined and Validated Device Model

Unified Device Model (UDM)

Chl 1: Manual Format Heterogeneity
Challenges for SDN Network Assimilation (SNA)

Vendor-specific Device Models (VDM)

Chl 1: Manual Format Heterogeneity

Preliminary Device Model

Chl 2: Errors & Ambiguity in Manuals

Refined and Validated Device Model

Unified Device Model (UDM)
Challenges for SDN Network Assimilation (SNA)

Vendor-specific Device Models (VDM)

Chl 1: Manual Format Heterogeneity

Preliminary Device Model

Chl 2: Errors & Ambiguity in Manuals

Refined and Validated Device Model

Chl 3: Heterogeneity between Configuration Models

Unified Device Model (UDM)
Our Key Contributions: NAssim
Our Key Contributions: NAssim

- NAssim Parser Framework (for Manual Format Heterogeneity)
  - Vendor-independent device model corpus format
  - Test-driven development procedure for reliable parsing
Our Key Contributions: NAAssim

• NAAssim Parser Framework (for Manual Format Heterogeneity)
  • Vendor-independent device model corpus format
  • Test-driven development procedure for reliable parsing

• NAAssim Validator (for Errors & Ambiguity in Manuals)
  • Validation on command-level, inter-command-level, and snippet-level
  • Identify 184 syntactic errors and 59 ambiguities in four mainstream vendors’ manuals
Our Key Contributions: NAssim

• NAssim Parser Framework (for Manual Format Heterogeneity)
  • Vendor-independent device model corpus format
  • Test-driven development procedure for reliable parsing

• NAssim Validator (for Errors & Ambiguity in Manuals)
  • Validation on command-level, inter-command-level, and snippet-level
  • Identify 184 syntactic errors and 59 ambiguities in four mainstream vendors’ manuals

• NAssim Mapper (for Heterogeneity between Configuration Models)
  • NetBERT: semantic similarity inference model for network configuration domain
  • NetBERT achieves 89% and 70% top 10 recall for mapping device models of Huawei and Nokia to a given UDM respectively
Our Key Contributions: NAssim

• NAssim Parser Framework (for Manual Format Heterogeneity)
  • Vendor-independent device model corpus format
  • Test-driven development procedure for reliable parsing

• NAssim Validator (for Errors & Ambiguity in Manuals)
  • Validation on command-level, inter-command-level, and snippet-level
  • Identify 184 syntactic errors and 59 ambiguities in four mainstream vendors’ manuals

• NAssim Mapper (for Heterogeneity between Configuration Models)
  • NetBERT: semantic similarity inference model for network configuration domain
  • NetBERT achieves 89% and 70% top 10 recall for mapping device models of Huawei and Nokia to a given UDM respectively

• Dataset: a validated and expert-curated dataset of parsed manual corpus for future research.
  (https://github.com/AmyWorkspace/nassim)
An Assistant Framework for Bridging the Last Mile Towards Centralized Network Configuration Management

Vendor
- Manual
- Configuration Files

SDN Assimilation
- Parser
- Manual Format Heterogeneity
- Preliminary Device Model
- Errors & Ambiguity in Manuals
- Refined and Validated Device Model
- CLI

SDN Controller
- Mapper
- Heterogeneity between Configuration Models
- Unified Device Model
SDN Network Assimilation (NAssim) in a Nutshell

Vendor
Manual

SDN Assimilation
Parser

Manual Format
Heterogeneity

Preliminary
Device Model

JSON

SDN Controller
Unified Device Model
NAssim Parser Framework: Key Insights
**NAssim Parser Framework: Key Insights**

### peer-as-number (BGP view)

**Function**
The peer-as-number command creates a peer and configures an AS number for a specified peer. The peer-as-number command advertises the AS number of a specified peer. By default, no BGP peer is configured, and no AS number is specified for a peer.

**Format**
```
peer ip-address as-number
undo peer ip-address
```

**Parameters**
- `ip-address`: Specifies the IPv4 address of a peer.
- `as-number`: Specifies an AS number.

**Examples**
- `# Set the AS number to 100 for IPv4 peer 10.1.1.1.
  peer 10.1.1.1 as-number 100
  peer 10.1.1.1 enable
  peer 10.1.1.1 remote-as 100
  peer 10.1.1.1 next-hop-local
  peer 10.1.1.1 route-map map1 in
  peer 10.1.1.1 route-map map2 out

### redistribute (BGP)

**Syntax**
```
redistribute (bgp)
```

**Description**
To inject routes from one routing domain into the Border Gateway Protocol (BGP), use the redistribute command. To remove the redistribution command from the configuration file and restore the system to its default condition in which the software does not redistribute routes, use the no form of this command.

**Syntax Description**
- `direct`:
  - Specifies that routes are directly connected on an interface.
- `rip instance-tag`:
  - Specifies routes from the RIP protocol. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters.
- `static`:
  - Specifies static routes.
- `route-map map-name`:
  - Specifies the identifier of a configured route map. Use a route map to filter which routes are redistributed into BGP.

**Examples**
- Specify the OSPF cost on the Interface as 33.
  ```
  [router-bgp]
  [interface-GigabitEthernet1/0]
  ospf cost 33
  ```

### label-ipv4

**Syntax**
```
label-ipv4
```

**Description**
The `label-ipv4` command configures an IPv4 address to be used for label-ipv4. By default, add-paths is not enabled for label-ipv4 IPv4 routes.

**Parameters**
- `context`:
  - The maximum number of labeled-unicast paths per IPv4 prefix that are allowed to be advertised. The actual number of advertised routes may be less, if the value is none, the router does not advertise any labeled-unicast paths.
- `unlabel-ipv4`:
  - The number of hops required to reach a labeled-unicast route per IPv4 prefix.
- `label-ipv4`:
  - The label used for labeled-unicast routes per IPv4 prefix.

**Examples**
- The maximum number of paths per labeled-unicast IPv4 prefix that are allowed to be advertised is add-paths.
  ```
  [interface-GigabitEthernet1/0]
  label-ipv4 unlabel-ipv4 10
  ```

**Related Commands**
- `default-metric (BGP)`
  - Sets the default metric for routes redistributed into BGP.
## NAssim Parser Framework: Key Insights

The same concept may have different names

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Vendor</th>
<th>Huawei</th>
<th>Cisco</th>
<th>Nokia</th>
<th>H3C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td></td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Format</code></td>
<td><code>&lt;class=&quot;pCE_CmdEnv&quot;&gt;</code></td>
<td><code>&lt;class=&quot;SyntaxHeader&quot;&gt;Syntax</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;Syntax</code></td>
</tr>
<tr>
<td>FuncDef</td>
<td></td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Function</code></td>
<td><code>&lt;class=&quot;pB1_Body1&quot;&gt;</code></td>
<td><code>&lt;class=&quot;DescriptionHeader&quot;&gt;Description</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;Description</code></td>
</tr>
<tr>
<td>ParentViews</td>
<td></td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Views</code></td>
<td><code>&lt;class=&quot;pCRCM_CmdRefCmdModes&gt;Command Modes</code></td>
<td><code>&lt;class=&quot;ContextHeader&quot;&gt;Context</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;View</code></td>
</tr>
<tr>
<td>ParaDef</td>
<td></td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Parameters</code></td>
<td><code>&lt;class=&quot;pCRSD_CmdRefSynDesc&quot;&gt;Syntax Description</code></td>
<td><code>&lt;class=&quot;ParametersHeader&quot;&gt;Parameters</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;Parameters</code></td>
</tr>
<tr>
<td>Examples</td>
<td></td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Examples</code></td>
<td><code>&lt;class=&quot;pCRE_CmdRefExample&quot;&gt;Examples</code></td>
<td>/</td>
<td><code>&lt;class=&quot;Command&quot;&gt;Examples</code></td>
</tr>
</tbody>
</table>

### Table 1: Diversity of Device User Manuals

The 'CLIs' field denotes the formal syntax of CLI commands, which are command templates with place-holder parameters and special characters to specify selection or optional branches. The 'ParaDef' field contains the implication and value range of place-holder parameters. The 'FuncDef' field describes the functionality of the complete CLI. The 'ParentViews' field indicates the parent/working views of CLIs, i.e., one CLI may have multiple viable working views. The 'Examples' field shows examples of common snippets, e.g., entering a parent view and issuing an instantiated CLI.
**NAssim Parser Framework: Key Insights**

Despite diverse styles, all manuals serve the same purpose: show how to configure the devices via CLI.

![Diagram](image)

**Table 1: Diversity of Device User Manuals**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Huawei</th>
<th>Cisco</th>
<th>Nokia</th>
<th>H3C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Format</code></td>
<td><code>&lt;class=&quot;pCE.CmdEnv&quot;&gt;</code></td>
<td><code>&lt;class=&quot;SyntaxHeader&quot;&gt;Syntax</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;Syntax</code></td>
</tr>
<tr>
<td>FuncDef</td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Function</code></td>
<td><code>&lt;class=&quot;pB1.Body1&quot;&gt;</code></td>
<td><code>&lt;class=&quot;DescriptionHeader&quot;&gt;Description</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;Description</code></td>
</tr>
<tr>
<td>ParentViews</td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Views</code></td>
<td><code>&lt;class=&quot;pCRCM.CmdRefCmdModes&quot;&gt;Command Modes</code></td>
<td><code>&lt;class=&quot;ContextHeader&quot;&gt;Context</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;View</code></td>
</tr>
<tr>
<td>ParaDef</td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Parameters</code></td>
<td><code>&lt;class=&quot;pCRSD.CmdRefSynDesc&quot;&gt;Syntax Description</code></td>
<td><code>&lt;class=&quot;ParametersHeader&quot;&gt;Parameters</code></td>
<td><code>&lt;class=&quot;Command&quot;&gt;Parameters</code></td>
</tr>
<tr>
<td>Examples</td>
<td><code>&lt;class=&quot;sectiontitle&quot;&gt;Examples</code></td>
<td><code>&lt;class=&quot;pCRE_CmdRefExample&quot;&gt;Examples</code></td>
<td>/</td>
<td><code>&lt;class=&quot;Command&quot;&gt;Examples</code></td>
</tr>
</tbody>
</table>

The 'CLIs' field denotes the formal syntax of CLI commands, which are command templates with place-holder parameters and special characters to specify selection or optional branches. The 'ParaDef' field contains the implication and value range of place-holder parameters. The 'FuncDef' field describes the functionality of the complete CLI. The 'ParentViews' field indicates the parent/working views of CLIs, i.e., one CLI may have multiple viable working views. The 'Examples' field shows examples of common snippets, e.g., entering a parent view and issuing an instantiated CLI.
Despite diverse styles, all manuals serve the same purpose: show how to configure the devices via CLI.
NAssim Parser Framework: Design

Parser

Abstract methods
parsing()
get_cli_parser()

Member methods
validating()

Validator

Parsing Completeness Validation

Parser_<vendor>

Member methods
parsing()
get_cli_parser()

Inherit

Manual

Preliminary VDM
NAssim Parser Framework: Design

### Keys

<table>
<thead>
<tr>
<th>Keys</th>
<th>Type Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>FuncDef</td>
<td>string</td>
</tr>
<tr>
<td>ParentViews</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>ParaDef</td>
<td>a list of dict (Keys:&quot;Paras&quot; and &quot;Info&quot;)</td>
</tr>
<tr>
<td>Examples</td>
<td>a list of list</td>
</tr>
</tbody>
</table>

Format Definition of Vendor-Independent Corpus (JSON)
NAssim Parser Framework: Design

**Parser**

- Abstract methods
  - parsing()
  - get_cli_parser()

- Member methods
  - parsing()
  - get_cli_parser()

**Validator**

- Parsing Completeness Validation

**Parser_<vendor>**

- Member methods
  - parsing()
  - get_cli_parser()

**Format Definition of Vendor-Independent Corpus (JSON)**

<table>
<thead>
<tr>
<th>Keys</th>
<th>Type Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>FuncDef</td>
<td>string</td>
</tr>
<tr>
<td>ParentViews</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>ParaDef</td>
<td>a list of dict (Keys:&quot;Paras&quot; and &quot;Info&quot;)</td>
</tr>
<tr>
<td>Examples</td>
<td>a list of list</td>
</tr>
</tbody>
</table>

Manual

Preliminary VDM
NAssim Parser Framework: Design

**Parser**

- Abstract methods
  - parsing()
  - get_cli_parser()

- Member methods
  - validating()

**Parser_<vendor>**

- Inherit

**Validator**

- Parsing Completeness Validation

<table>
<thead>
<tr>
<th>Keys</th>
<th>Type Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>FuncDef</td>
<td>string</td>
</tr>
<tr>
<td>ParentViews</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>ParaDef</td>
<td>a list of dict (Keys: &quot;Paras&quot; and &quot;Info&quot;)</td>
</tr>
<tr>
<td>Examples</td>
<td>a list of list</td>
</tr>
</tbody>
</table>

Format Definition of Vendor-Independent Corpus (JSON)
Format Definition of Vendor-Independent Corpus (JSON)

<table>
<thead>
<tr>
<th>Keys</th>
<th>Type Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>FuncDef</td>
<td>string</td>
</tr>
<tr>
<td>ParentViews</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>ParaDef</td>
<td>a list of dict (Keys:&quot;Paras&quot; and &quot;Info&quot;)</td>
</tr>
<tr>
<td>Examples</td>
<td>a list of list</td>
</tr>
</tbody>
</table>

NAssim Parser Framework: Design

Validator

Parser_<vendor>

Parser

Manual

Preliminary VDM

A sample of parsed VDM corpus.
Vendor-independent parsed format captures the commonality of manuals from different vendors, also balancing extendibility and human-readability.

<table>
<thead>
<tr>
<th>Keys</th>
<th>Type Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIs</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>FuncDef</td>
<td>string</td>
</tr>
<tr>
<td>ParentViews</td>
<td>a list of string (non-empty list)</td>
</tr>
<tr>
<td>ParaDef</td>
<td>a list of dict (Keys: &quot;Paras&quot; and &quot;Info&quot;)</td>
</tr>
<tr>
<td>Examples</td>
<td>a list of list</td>
</tr>
</tbody>
</table>

Format Definition of Vendor-Independent Corpus (JSON)

A sample of parsed VDM corpus.
SDN Network Assimilation (NAssim) in a Nutshell

Vendor

Manual

SDN Assimilation

Parser

Manual Format
Heterogeneity

Preliminary
Device Model

SDN Controller

Unified Device Model

Vendor

Manual

SDN Assimilation

Parser

Manual Format
Heterogeneity

Preliminary
Device Model

SDN Controller

Unified Device Model
SDN Network Assimilation (NAssim) in a Nutshell

Vendor

Parser

Manual Format Heterogeneity

Validator

Errors & Ambiguity in Manuals

SDN Assimilation

JSON

Preliminary Device Model

SDN Controller

CLI

Refined and Validated Device Model

Unified Device Model

Manual

Configuration Files
NAssim Validator: Key Insights
Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.
Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.

Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.

NAssim Validator: Key Insights

Syntactic Ambiguities

neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> } ]

Correction Option 1: removing the left bracket
neighbor { <ip-addr> | <ip-prefix/length> } remote-as { <as-num> [ <.as-num> ] | route-map <name> }

Correction Option 2: adding a right bracket after remote-as symbol
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as ] { <as-num> [ <.as-num> ] | route-map <name> }

Correction Option 3: adding a right bracket at the end of the CLI
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> } ]

An example of ambiguous CLI command template*


Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.
NAssim Validator: Key Insights

**Syntactic Ambiguities**

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> }[ remote-as { <as-num> [ <.as-num> ] | route-map <name> }
```

*unpaired left bracket*

**Correction Option 1: removing the left bracket**

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> } remote-as { <as-num> [ <.as-num> ] | route-map <name> }
```

**Correction Option 2: adding a right bracket after remote-as symbol**

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> }[ remote-as ]{ <as-num> [ <.as-num> ] | route-map <name> }
```

**Correction Option 3: adding a right bracket at the end of the CLI**

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> }[ remote-as { <as-num> [ <.as-num> ] | route-map <name> }]
```

**Hierarchy Ambiguities**

**An example of ambiguous CLI command template***

---

**An example of ambiguous view**

---

**Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.**

---

Syntactic Ambiguities

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> }
```

Correction Option 1: removing the left bracket

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> } remote-as { <as-num> [ <.as-num> ] | route-map <name> }
```

Correction Option 2: adding a right bracket after remote-as symbol

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as ] { <as-num> [ <.as-num> ] | route-map <name> }
```

Correction Option 3: adding a right bracket at the end of the CLI

```plaintext
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as <as-num> [ <.as-num> ] | route-map <name> ]
```

An example of ambiguous CLI command template*

Hierarchy Ambiguities

```plaintext
{  
"CLIs": [  
"import-source acl { acl-number | acl-name }",  
"undo import-source"  
],  
"...": "...",  
"ParentView": ["VPN instance MSDP view", "MSDP view of a public network instance"],  
"...": "...",  
"Examples": [  
["<HUAWEI> system-view",  
["<HUAWEI> acl number 3101",  
"...",  
["*HUAWEI-acl4-advance-3101] quit",  
["*HUAWEI] multicast routing-enable",  
["*HUAWEI] msdp",  
["*HUAWEI-msdp] import-source acl 3101"]  
]  
}
```

An example of ambiguous view

Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.

Syntactic Ambiguities

```
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> }
```

**Correction Option 1: removing the left bracket**
```
neighbor { <ip-addr> | <ip-prefix/length> } remote-as { <as-num> [ <.as-num> ] | route-map <name> }
```

**Correction Option 2: adding a right bracket after remote-as symbol**
```
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as ] { <as-num> [ <.as-num> ] | route-map <name> }
```

**Correction Option 3: adding a right bracket at the end of the CLI**
```
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> } ]
```

An example of ambiguous CLI command template*

Hierarchy Ambiguities

```
{ "CLIs": [ "import-source acl { acl-number | acl-name }", "undo import-source" ], ...
"ParentView": ["VPN instance MSDP view", "MSDP view of a public network instance"], ...
```

An example of ambiguous view

**Manuals are not fully reliable. A rigorous validation scheme needs to catch inevitable errors and ambiguities in human-written manuals.**

NAssim Validator: Design

Validator

1. **Formal Syntax Validation**
   - Preliminary VDM (CLI syntax validated)

2. **Hierarchy Derivation & Validation**
   - Preliminary VDM
   - Semantics

3. **Device Configuration Validation**
   - Refined VDM
   - Semantics
   - Device Configurations

Refined VDM (Empirically-validated)
NAssim Validator: Design

Validator

1. Formal Syntax Validation

Preliminary VDM

Preliminary VDM

(JSON)

(JSON)

Preliminary VDM (CLI syntax validated)
NAssim Validator: Design

Validator

1. Formal Syntax Validation

Preliminary VDM

Preliminary VDM

(CLI syntax validated)

Express command conventions into their equivalent Backus Normal Form (BNF) for formal validation.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface font</strong></td>
<td>Commands and keywords are in boldface.</td>
</tr>
<tr>
<td><strong>italic font</strong></td>
<td>Arguments for which you supply values are in italics.</td>
</tr>
<tr>
<td><code>[]</code></td>
<td>Elements in square brackets are optional.</td>
</tr>
<tr>
<td>`{x</td>
<td>y</td>
</tr>
<tr>
<td>`[x</td>
<td>y</td>
</tr>
<tr>
<td>string</td>
<td>A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.</td>
</tr>
</tbody>
</table>

Command convention of Cisco manuals

```python
import pyparsing as p

# syntax parser for Cisco CLI
word = p.Word(p.printables, exclude_chars='{{|\|\n').setParseAction(leaf_gen)
ele = p.Forward()
items = ele + p.ZeroOrMore(' | ' + ele)
select = p.Group('[' + items + ']').setParseAction(select_gen)
option = p.Group('[' + items + ']').setParseAction(option_gen)
ele << p.OneOrMore(option + select + word).setParseAction(ele_gen)
syntax_parser = ele
```

Code snippet for syntax parser generation
NAssim Validator: Design

 Validator

  1 Formal Syntax Validation

 Preliminary VDM (CLI syntax validated)
NAssim Validator: Design

Validator

1. Formal Syntax Validation
2. Hierarchy Derivation & Validation

Preliminary VDM

Preliminary VDM (CLI syntax validated)

Refined VDM

JSON

Semantic

CLI
NAssim Validator: Design

Validator

1. Formal Syntax Validation
2. Hierarchy Derivation & Validation

Preliminary VDM

Preliminary VDM (CLI syntax validated)

Refined VDM

Leverage example snippets to derive the relationship between CLI commands, i.e., CLI model hierarchy.
Leverage example snippets to derive the relationship between CLI commands, i.e., CLI model hierarchy.
NAssim Validator: Design

1. **Formal Syntax Validation**
   - Preliminary VDM
   - Preliminary VDM (CLI syntax validated)

2. **Hierarchy Derivation & Validation**
   - Refined VDM

{ "CLIs": [ "peer <ipv4-address> group <group-name>", "undo peer <ipv4-address> group <group-name>"] },
...
"ParentView": ["BGP view"],
...
"Examples": [ "<HUAWEI> system-view", "[~HUAWEI-bgp 100", "["HUAWEI-peer 10.1.1.1 group test"] ]
}

Leverage example snippets to derive the relationship between CLI commands, i.e., CLI model hierarchy.
NAssim Validator: Design

Validator

1. Formal Syntax Validation
2. Hierarchy Derivation & Validation

Preliminary VDM (CLI syntax validated) → Preliminary VDM → Refined VDM
NAssim Validator: Design

 validator

1. Formal Syntax Validation
2. Hierarchy Derivation & Validation

- Preliminary VDM
- Preliminary VDM (CLI syntax validated)
- Refined VDM
- JSON
- Syntax Validation

An example ambiguous view not be fully resolved by our hierarchy derivation scheme

Quantify the certainty of the derivation to facilitate expert intervention
NAssim Validator: Design

Validator

1. Formal Syntax Validation

2. Hierarchy Derivation & Validation

Preliminary VDM (CLI syntax validated)

Refined VDM
NAssim Validator: Design

Validator

1. **Formal Syntax Validation**
   - Preliminary VDM
     - JSON
   - Preliminary VDM (CLI syntax validated)

2. **Hierarchy Derivation & Validation**
   - JSON
   - Semantics
   - Refined VDM
     - CLI

3. **Device Configuration Validation**
   - JSON
   - Semantics
   - Device Configurations
NAssim Validator: Design

Device Configurations

Part of the VDM

Preliminary VDM (CLI syntax validated)

Semantics

Refined VDM

Device Configurations

Correctness Guarantee

1. bgp 55990
   router-id 172.16.0.28
   ... timer keepalive 10 hold 30 peer 10.147.5.124 as-number 55990
   ... ipv4-family unicast
   preference 80 140 200
   maximum load-balancing 8
   ... route-policy RR-IN permit node 1000
   if-match acl 2000
   ...
NAssim Validator: Design

Validator

1. Formal Syntax Validation
2. Hierarchy Derivation & Validation
3. Device Configuration Validation

Preliminary VDM

Device Configurations

Correctness Guarantee
## VDM Construction Phase: Parser + Validator

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Statistics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CLI Commands</td>
<td>12874</td>
<td>278</td>
<td>14046</td>
<td>759</td>
<td></td>
</tr>
<tr>
<td>#Views</td>
<td>607</td>
<td>27</td>
<td>3832</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>#CLI-View Pairs</td>
<td>36274</td>
<td>366</td>
<td>22734</td>
<td>851</td>
<td></td>
</tr>
<tr>
<td><strong>Adaption Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parsing() LOC</td>
<td>45</td>
<td>52</td>
<td>57</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>get_cli_parser() LOC</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Syntax Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Invalid CLI Commands</td>
<td>13</td>
<td>19</td>
<td>139</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td><strong>Model Hierarchy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Derivation &amp; Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Example Snippets</td>
<td>15466</td>
<td>523</td>
<td>/</td>
<td>1147</td>
<td></td>
</tr>
<tr>
<td>Construction Time (second)</td>
<td>785.58</td>
<td>14.29</td>
<td>94.56*</td>
<td>34.3</td>
<td></td>
</tr>
<tr>
<td>#Ambiguous Views</td>
<td>47</td>
<td>8</td>
<td>/</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Device Configuration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Config Files</td>
<td>197</td>
<td>/</td>
<td>416</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Matching Ratio</td>
<td>100%</td>
<td>/</td>
<td>100%</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3:** Evaluation of the VDM Construction Phase. *Nokia manuals do not provide examples, but they explicitly specify model hierarchy in the manuals. Thus, we extract the hierarchy using `Parser_<nokia>` by implementing extra functions.*
### VDM Construction Phase: Parser + Validator

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Statistics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CLI Commands</td>
<td>12874</td>
<td>278</td>
<td>14046</td>
<td>759</td>
<td></td>
</tr>
<tr>
<td>#Views</td>
<td>607</td>
<td>27</td>
<td>3832</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>#CLI-View Pairs</td>
<td>36274</td>
<td>366</td>
<td>22734</td>
<td>851</td>
<td></td>
</tr>
<tr>
<td><strong>Adaption Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parsing() LOC</td>
<td>45</td>
<td>52</td>
<td>57</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>get_cli_parser() LOC</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Syntax Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Invalid CLI Commands</td>
<td>13</td>
<td>19</td>
<td>139</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td><strong>Model Hierarchy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Derivation &amp; Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Example Snippets</td>
<td>15466</td>
<td>523</td>
<td>/</td>
<td>1147</td>
<td></td>
</tr>
<tr>
<td>Construction Time (second)</td>
<td>785.58</td>
<td>14.29</td>
<td>94.56*</td>
<td>34.3</td>
<td></td>
</tr>
<tr>
<td>#Ambiguous Views</td>
<td>47</td>
<td>8</td>
<td>/</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Device Configuration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Config Files</td>
<td>197</td>
<td>/</td>
<td>416</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Matching Ratio</td>
<td>100%</td>
<td>/</td>
<td>100%</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3:** Evaluation of the VDM Construction Phase. *Nokia manuals do not provide examples, but they explicitly specify model hierarchy in the manuals. Thus, we extract the hierarchy using **Parse <nokia>** by implementing extra functions.
## VDM Construction Phase: Parser + Validator

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Statistics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CLI Commands</td>
<td>12874</td>
<td>278</td>
<td>14046</td>
<td>759</td>
<td></td>
</tr>
<tr>
<td>#Views</td>
<td>607</td>
<td>27</td>
<td>3832</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>#CLI-View Pairs</td>
<td>36274</td>
<td>366</td>
<td>22734</td>
<td>851</td>
<td></td>
</tr>
<tr>
<td><strong>Adaption Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parsing() LOC</td>
<td>45</td>
<td>52</td>
<td>57</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>get_cli_parser() LOC</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Syntax Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Invalid CLI Commands</td>
<td>13</td>
<td>19</td>
<td>139</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td><strong>Model Hierarchy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Derivation &amp; Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Example Snippets</td>
<td>15466</td>
<td>523</td>
<td>/</td>
<td>1147</td>
<td></td>
</tr>
<tr>
<td>Construction Time (second)</td>
<td>785.58</td>
<td>14.29</td>
<td>94.56*</td>
<td>34.3</td>
<td></td>
</tr>
<tr>
<td>#Ambiguous Views</td>
<td>47</td>
<td>8</td>
<td>/</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Device Configuration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Config Files</td>
<td>197</td>
<td>/</td>
<td>416</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Matching Ratio</td>
<td>100%</td>
<td>/</td>
<td>100%</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Evaluation of the VDM Construction Phase. *Nokia manuals do not provide examples, but they explicitly specify model hierarchy in the manuals. Thus, we extract the hierarchy using Parser_<nokia> by implementing extra functions.
### VDM Construction Phase: Parser + Validator

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Statistics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CLI Commands</td>
<td>12874</td>
<td>278</td>
<td>14046</td>
<td>759</td>
</tr>
<tr>
<td>#Views</td>
<td>607</td>
<td>27</td>
<td>3832</td>
<td>28</td>
</tr>
<tr>
<td>#CLI-View Pairs</td>
<td>36274</td>
<td>366</td>
<td>22734</td>
<td>851</td>
</tr>
<tr>
<td><strong>Adaption Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parsing() LOC</td>
<td>45</td>
<td>52</td>
<td>57</td>
<td>41</td>
</tr>
<tr>
<td>get_cli_parser() LOC</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td><strong>Syntax Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Invalid CLI Commands</td>
<td>13</td>
<td>19</td>
<td>139</td>
<td>13</td>
</tr>
<tr>
<td><strong>Model Hierarchy Derivation &amp; Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Example Snippets</td>
<td>15466</td>
<td>523</td>
<td>/</td>
<td>1147</td>
</tr>
<tr>
<td>Construction Time (second)</td>
<td>785.58</td>
<td>14.29</td>
<td>94.56*</td>
<td>34.3</td>
</tr>
<tr>
<td>#Ambiguous Views</td>
<td>47</td>
<td>8</td>
<td>/</td>
<td>4</td>
</tr>
<tr>
<td><strong>Device Configuration Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Config Files</td>
<td>197</td>
<td>/</td>
<td>416</td>
<td>/</td>
</tr>
<tr>
<td>Matching Ratio</td>
<td>100%</td>
<td>/</td>
<td>100%</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 3: Evaluation of the VDM Construction Phase. *Nokia manuals do not provide examples, but they explicitly specify model hierarchy in the manuals. Thus, we extract the hierarchy using Parser_<nokia> by implementing extra functions.

Refer to our repo for more details: [https://github.com/AmyWorkspace/nassim](https://github.com/AmyWorkspace/nassim)
SDN Network Assimilation (NAssim) in a Nutshell

Vendor

SDN Assimilation

Parser

Manual Format Heterogeneity

Preliminary Device Model

Validator

Errors & Ambiguity in Manuals

Refined and Validated Device Model

SDN Controller

Configuration Files

CLI

Unified Device Model

Manual

Parser

JSON

semantics

CLI

JSON

Vendor Manual

SDN Assimilation

Parser

Manual Format Heterogeneity

Preliminary Device Model

Validator

Errors & Ambiguity in Manuals

Refined and Validated Device Model

SDN Controller

Configuration Files

CLI

Unified Device Model
SDN Network Assimilation (NAssim) in a Nutshell

Vendor
- Manual
- Configuration Files

SDN Assimilation
- Parser
- Validator
- Manual Format Heterogeneity
  - Preliminary Device Model
  - Errors & Ambiguity in Manuals
  - Refined and Validated Device Model

SDN Controller
- Mapper
  - Heterogeneity between Configuration Models
  - mapping
  - Unified Device Model

Manual Format Heterogeneity
- CLI
- semantics
NAssim Mapper: Key Insights

manual

Unified Device Model (UDM)

BGP

AS Number

BFD Rx

BFD Multiplier

BFD Session
-Link Bundle

BFD

Session

Link Bundle
**NAssim Mapper: Key Insights**

**CLI command template**

```
system-view
lag <lag-id>
bfd
  family { ipv4 | ipv6 }
multipihier [ <multiplier> ]
```

**Handcrafted Mapping**

Read manuals

**Unified Device Model (UDM)**

- BGP
- AS Number
- BFD Rx
- BFD Session
- Link Bundle
- BFD Multiplier

Mapping Expert
The key of SNA is to pair semantically similar configuration items.
**NAssim Mapper: Key Insights**

The key of SNA is to pair semantically similar configuration items.

---

**CLI command template**

```
system-view
  lag <lag-id>
  bfd
    family { ipv4 | ipv6 }
    multiplier [ <multiplier> ]
```

---

**Handcrafted Mapping**

**Vendor-specific Device Model (VDM)**

```
config
  { }
```

---

**Unified Device Model (UDM)**

---

**The key of SNA is to pair semantically similar configuration items.**
The key of SNA is to pair semantically similar configuration items.
NAssim Mapper: NetBERT
NAssim Mapper: NetBERT
NAssim Mapper: NetBERT

Context

Mapper: NetBERT

VDM

semantics

{}

JSON

CLI

UDM

Context

?
For VDM (or UDM) $M$, with $n_M$ parameters and one of its parameter $p_i^M$ ($i \in 0,1,...,n_M - 1$), its context is

$$C(p_i^M) = [s_i^0, s_i^1, ..., s_i^{k_M-1}]$$

where each $s_i$ is a text sequence and $k_M$ is the number of extracted sequences.
NAssim Mapper: NetBERT

Context

Mapper: NetBERT

VDM

semantics

{} JSON

CLI

UDM

Context

?
NAssim Mapper: NetBERT

Mapper: NetBERT

Context

NetBERT

Embedding

NetBERT

Context

VDM

semantics

{}: JSON

CLI

UDM

CONTEXT MAPPER: NetBERT
Encode each of the text separately and then produce an embedding matrix. Assume the output dimension of the encoder $e(\cdot)$ is $m$, the context embedding of $p_i^M$ is:

$$E_i^M = e(C(p_i^M)) = e([s_i^0, s_i^1, ..., s_i^{K-1}]) \in \mathbb{R}^{K \times m}$$
NAssim Mapper: NetBERT

Context

Mapper: NetBERT

NetBERT

Embedding

NetBERT

Context

VDM

semantics

{}:} JSON

CLI

UDM

mapper: NetBERT

Context
For $E_i^V$ and $E_j^U$ with the number of contexts $k_V$ and $k_U$, the similarity score for context embedding is:

$$\text{Sim}(E_i^V, E_j^U) = w \cdot (E_i^V \otimes E_j^U)$$

where $w = \langle w_0, w_1, \ldots, w_{k_U \times k_U - 1} \rangle$ (s.t. $\sum_{t=0}^{k_U \times k_U - 1} w_t = 1$)

$\otimes$ : compute row-wise cosine similarities of two matrices
NAssim Mapper: NetBERT

Mapper: NetBERT

Context

VDM

semantics

JSON

CLI

NetBERT

Embedding

Cosine Similarities

Score

UDM

Context
NAssim Mapper: NetBERT

Mapper: NetBERT

Context

NetBERT

NetBERT

Cosine Similarities

Embedding

Score

VDM

UDM

semantics

{} {} JSON

CLI

Top 5 semantically similar

Ranking:
NAssim Mapper: NetBERT Training

Mapper: NetBERT

VDM Context  UDM Context'

NetBERT  NetBERT

Embedding

Cosine Similarities

SCORE
NAssim Mapper: NetBERT Training


Pre-trained model*
Pre-trained model*

Net-Config Domain Data

Mapper: NetBERT

VDM Context

NetBERT

Embedding

Cosine Similarities

UDM Context'

NetBERT

Score

NAssim Mapper: NetBERT Training

**NAssim Mapper: NetBERT Training**

*Pre-trained model*

**Domain Adaptation**

**Net-Config Domain Data**

**Negative Sampling**

UDM-VDM mappings(+)

UDM-VDM mappings(-)

**Mapper: NetBERT**

VDM Context

UDM Context'

NetBERT

NetBERT

Embedding

Cosine Similarities

## VDM-UDM Mapping Phase: Mapper

<table>
<thead>
<tr>
<th>Mapping Setting</th>
<th>Models</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huawei-UDM</td>
<td>IR</td>
<td>41</td>
<td>52</td>
<td>61</td>
<td>66</td>
<td>69</td>
<td>74</td>
<td>76</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>90</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>40</td>
<td>53</td>
<td>59</td>
<td>63</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>72</td>
<td>77</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>53</td>
<td>66</td>
<td>72</td>
<td>76</td>
<td>79</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>84</td>
<td>85</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>43</td>
<td>61</td>
<td>68</td>
<td>74</td>
<td>75</td>
<td>77</td>
<td>79</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>56</td>
<td>69</td>
<td>75</td>
<td>79</td>
<td>81</td>
<td>83</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>91</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>57</td>
<td>69</td>
<td>74</td>
<td>78</td>
<td>80</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>89</td>
<td>91</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>58</td>
<td>71</td>
<td>78</td>
<td><strong>81</strong></td>
<td>83</td>
<td>85</td>
<td><strong>86</strong></td>
<td>87</td>
<td><strong>88</strong></td>
<td>89</td>
<td>93</td>
<td>95</td>
</tr>
<tr>
<td>Nokia-UDM</td>
<td>IR</td>
<td>24</td>
<td>31</td>
<td>41</td>
<td>45</td>
<td>48</td>
<td>56</td>
<td>57</td>
<td>59</td>
<td>59</td>
<td>60</td>
<td>66</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>20</td>
<td>27</td>
<td>31</td>
<td>33</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>42</td>
<td>45</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>34</td>
<td>35</td>
<td>38</td>
<td>44</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>52</td>
<td>52</td>
<td>52</td>
<td>58</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>24</td>
<td>31</td>
<td>35</td>
<td>40</td>
<td>42</td>
<td>43</td>
<td>46</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>57</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>34</td>
<td>40</td>
<td>42</td>
<td><strong>49</strong></td>
<td>52</td>
<td>52</td>
<td><strong>54</strong></td>
<td>55</td>
<td><strong>55</strong></td>
<td>58</td>
<td>62</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>34</td>
<td>40</td>
<td>43</td>
<td>50</td>
<td><strong>53</strong></td>
<td><strong>58</strong></td>
<td>66</td>
<td>67</td>
<td>67</td>
<td>70</td>
<td>71</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td><strong>35</strong></td>
<td><strong>41</strong></td>
<td><strong>47</strong></td>
<td><strong>51</strong></td>
<td><strong>55</strong></td>
<td><strong>57</strong></td>
<td>65</td>
<td><strong>67</strong></td>
<td><strong>68</strong></td>
<td>68</td>
<td><strong>71</strong></td>
<td><strong>73</strong></td>
</tr>
</tbody>
</table>
### VDM-UDM Mapping Phase: Mapper

<table>
<thead>
<tr>
<th>Mapping Setting</th>
<th>Models</th>
<th>k in recall@top k (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 20 30</td>
</tr>
<tr>
<td>Huawei-UDM</td>
<td>IR</td>
<td>41 52 61 66 69 74 76 78 79 80 90 93</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>40 53 59 63 66 67 68 69 70 72 77 81</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>53 66 72 76 79 80 81 82 84 85 89 92</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>43 61 68 74 75 77 79 80 81 82 89 92</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>56 69 75 79 81 83 85 86 87 88 91 94</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>57 69 74 78 80 84 85 86 87 88 91 94</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>58 71 78 81 83 85 86 87 88 89 93 95</td>
</tr>
<tr>
<td>Nokia-UDM</td>
<td>IR</td>
<td>24 31 41 45 48 56 57 59 59 60 66 70</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>20 27 31 33 37 38 39 39 39 42 45 48</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>34 35 38 44 49 49 49 52 52 52 58 53</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>24 31 35 40 42 43 46 48 48 48 57 61</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>34 40 42 49 52 52 54 55 55 58 62 72</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>34 40 43 50 53 58 66 67 67 70 71 73</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>35 41 47 51 55 57 65 67 68 68 71 73</td>
</tr>
</tbody>
</table>

Recall@top-k denotes the percentage of test cases where the correct matching parameters are in top k recommendation by Mapper.
VDM-UDM Mapping Phase: Mapper

<table>
<thead>
<tr>
<th>Mapping Setting</th>
<th>Models</th>
<th>k in recall@top k (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Huawei-UDM</td>
<td>IR</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>58</td>
</tr>
<tr>
<td>Nokia-UDM</td>
<td>IR</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>35</td>
</tr>
</tbody>
</table>

Recall@top-k denotes the percentage of test cases where the correct matching parameters are in top k recommendation by Mapper.
## VDM-UDM Mapping Phase: Mapper

<table>
<thead>
<tr>
<th>Mapping Setting</th>
<th>Models</th>
<th>k in recall@top k (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Huawei-UDM</td>
<td>IR</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>58</td>
</tr>
<tr>
<td>Nokia-UDM</td>
<td>IR</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>35</td>
</tr>
</tbody>
</table>

Recall@top-k denotes the percentage of test cases where the correct matching parameters are in top k recommendation by Mapper.
Key Takeways
Key Takeways

- Soft-defined network assimilation (SNA) is essential for managing multi-vendor network. Our solution NAssim seeks to transform tedious and error-prone SNA process to automated and efficient manner.
Key Takeways

• Soft-defined network assimilation (SNA) is essential for managing multi-vendor network. Our solution NAssim seeks to transform tedious and error-prone SNA process to automated and efficient manner.

• Configuration manuals, as human-written documents are not fully reliable, including inevitable errors and ambiguities.
Key Takeways

• Soft-defined network assimilation (SNA) is essential for managing multi-vendor network. Our solution NAssim seeks to transform tedious and error-prone SNA process to automated and efficient manner.

• Configuration manuals, as human-written documents are not fully reliable, including inevitable errors and ambiguities.

• NAssim features a unified parser framework, a rigorous validator and a mapper using the domain-adapted BERT model to produce human-comprehensible recommended mapping between the validated configuration model and the one in the SDN controller.
**Key Takeways**

- Soft-defined network assimilation (SNA) is essential for managing multi-vendor network. Our solution NAssim seeks to transform tedious and error-prone SNA process to automated and efficient manner.

- Configuration manuals, as human-written documents are not fully reliable, including inevitable errors and ambiguities.

- NAssim features a unified parser framework, a rigorous validator and a mapper using the domain-adapted BERT model to produce human-comprehensible recommended mapping between the validated configuration model and the one in the SDN controller.

- We release a validated and expert-curated dataset of parsed manual corpus for future research. ([https://github.com/AmyWorkspace/nassim](https://github.com/AmyWorkspace/nassim))
Software-defined Network Assimilation: Bridging the Last Mile Towards Centralized Network Configuration Management with NAssim

Huangxun Chen, Yukai Miao, Li Chen, Haifeng Sun, Hong Xu, Libin Liu, Gong Zhang, Wei Wang

Thank you!

Q&A

https://amyworkspace.github.io/hxchen/
https://github.com/AmyWorkspace/nassim